universität wien

DISSERTATION / DOCTORAL THESIS

Titel der Dissertation / Title of the Doctoral Thesis

„Knowledge-Supervised Information Extraction for Natural Language Processing“

verfasst von / submitted by

Luisa März B.Sc. MSc.

angestrebter akademischer Grad / in partial fulfilment of the requirements for the degree of

Doktorin der technischen Wissenschaften (Dr. techn.)

Wien, 2022 / Vienna, 2022

Studienkennzahl lt. Studienblatt / degree programme code as it appears on the student record sheet:	UA 786 880
Studienrichtung lt. Studienblatt / degree programme as it appears on the student record sheet:	Informatik
Betreut von / Supervisor:	Univ.-Prof. Dr.-Ing. Benjamin Roth, B.Sc. M.Sc.
Betreut von / Supervisor:	Prof. Dr. Hinrich Schütze

Acknowledgements

During the course of my doctorate, various people have accompanied me - some all the time, some for a shorter time. I am incredibly grateful for the valuable support from every single one and I would like to highlight and thank in particular:

Univ.-Prof. Dr.-Ing. Benjamin Roth - My greatest thanks go to my supervisor, who accompanied me for the whole three and a half years. Ben, I want to thank you for your constant support, feedback, and all discussions. I don't think I could have received such great and valuable supervision from anyone else. You motivated me right from the start and brought out the best in me (not only) through critical feedback. Even if this work is about unsupervised learning, I was lucky enough to have a particularly individual supervised learning through you.

Prof. Dr. Hinrich Schütze - I would also like to thank my second supervisor, who supported me especially in the first one and a half years. Hinrich, I valued the joint conversations and discussions, as well as the support with my first doctoral project and the opportunity to start my doctorate together.

Dr. Christoph Ringlstetter - I also thank my supervisor in industry and team lead. Christoph, after my spontaneous adoption into the team, you continuously supported and accompanied me. I am very grateful that you managed to steer me into productive paths, to encourage me and at the same time to slow down in the right places.

Univ.-Prof. Dipl.-Inform. Univ. Dr. Claudia Plant and Univ.-Prof. Dr. Michael Wiegand - Thank you for your willingness to read and review my dissertation.

Stefan, you not only worked with me on several publications, but really had an open ear for every question and helped me so many times. Thank you!

Thank you **Andy, Christian, Dietrich, Ehsan, Erion, Fabienne, Franziska, Katharina and Nina** for researching and discussing approaches together and for working on publishing our work. It was a pleasure to work with you all and I appreciated your collaboration immensely!

Thanks to **Anastasiia, Pedro and Vasiliki** for providing valuable feedback on some of my publications and on this thesis. Thank you for showing me further perspectives and for sweetening the writing phase with critical, appreciative, and humorous comments!

Finally, I would like to thank the greatest support of all – my husband **Flo**. Without you, I wouldn't even have started my studies and I can't even put into words how much your support, motivation, and backing have meant to me over the last few years. Thank you so much for enduring my madness!
My special thank you goes to my son **Matteo**, who sometimes more and sometimes less understood that I had to look for bugs in the computer rather than in the garden.

Abstract

This work approaches knowledge supervised information extraction for natural language processing. More precisely, it is about how knowledge can be integrated into supervised machine learning models (in this case neural networks) for information extraction. Machine learning models are usually tailored to generalize as much as possible in their learning process to offer a wide range of applications. In cases where generalized models do not sufficiently learn specialized tasks, the integration of knowledge is particularly important and is therefore investigated in this thesis. In particular, three different perspectives are highlighted: i) data-centric knowledge supervision, ii) model-centric knowledge supervision, and iii) knowledge supervision for industrial information extraction.

First, machine learning in general, knowledge supervised machine learning, and representation learning, as well as downstream tasks in natural language processing are explained. The above mentioned perspectives on knowledge supervision are introduced and it is explained how knowledge about domain, task, and/or method can be integrated into machine learning models.

The contributing articles for data-centric knowledge supervision tackle named entity recognition in historical text, a domain with peculiarities that differ from standard contemporary texts many machine learning models are trained on. It is shown that data-centric knowledge integration, for example via pre-training domain-specific language models or the creation of knowledge-enriched word representations, is beneficial and enhances model performance. The findings are confirmed by the results across different data sets and experiments.

One article on model-centric knowledge supervision proposes an adversarial neural network for weakly supervised data that aims to learn robust and noise-invariant input representations. Based on three classification datasets, it is shown that expanding the model focus to all relevant information contained in input instances improves downstream performance of machine learning models. Another model-centric approach measures generalization of models trained on misleading noisy data obtained from weak supervision. Results show that the adversarial network is able to control the degree of generalizing from noisy signals. A comparison of different state-of-the-art models for weak supervision shows that generalization and model performance are not related one-to-one.

The knowledge supervision approach for industrial information extraction establishes best-practices and provides a guide on how to turn rule-based systems into machine learning models by utilizing weak supervision. Along with a case study, the process is explained in detail and illustrated by examples.

The second part of this work includes all the original manuscripts of the contributing articles, cites the respective source, and presents the contributions of each author.

Kurzfassung

Diese Arbeit befasst sich mit der wissensüberwachten Informationsextraktion im Bereich des *Natural Language Processing*. Es geht darum, wie Wissen in überwachte Lernmodelle zur Informationsextraktion (in diesem Fall neuronale Netze), integriert werden kann. Maschinelle Lernmodelle sind darauf zugeschnitten, in ihrem Lernprozess möglichst gut verallgemeinern. In Fällen, in denen generalisierte Modelle daran scheitern, Spezialaufgaben zufriedenstellend zu lernen, ist die Integration von Wissen besonders wichtig. Drei Perspektiven werden in dieser Arbeit besonders hervorgehoben: i) datenzentrierte und ii) modellzentrierte Wissensüberwachung, sowie iii) Wissensüberwachung für die Informationsextraktion in der Industrie.
Zunächst werden maschinelles und wissensüberwachtes maschinelles Lernen, das Lernen von Repräsentationen und Anwendungen für *Natural Language Processing* erklärt. Die drei Perspektiven werden eingeführt und es wird erklärt, wie Wissen über Domäne, Task und/oder Methode in maschinelle Lernmodelle integriert werden kann.
Die Beiträge zur datenzentrierten Wissensüberwachung befassen sich mit Eigennamenerkennung in historischen Texten, einer speziellen Domäne, die sich von zeitgenössischen Standardtexten unterscheidet. Es wird gezeigt, dass die datenzentrierte Wissensintegration, bspw. über vor-trainierte domänenspezifische Sprachmodelle oder die Erstellung von mit Wissen angereicherten Wortrepräsentationen, die Modellleistung steigert. Die Erkenntnisse werden durch Ergebnisse auf mehreren Datensätzen bestätigt.
Einer der Beiträge zur modellzentrierten Wissensüberwachung präsentiert ein neuronales Netzwerk, das im Inneren mit zwei konkurrierenden neuronalen Netzen arbeitet (*adversarial network*). Dieses ist für die Verarbeitung von *Weak Supervision* Daten gedacht und lernt robuste und rauschinvariante Eingaberepräsentationen. Es wird gezeigt, dass die Erweiterung des Modellfokus auf alle relevanten Informationen, die in Eingabeinstanzen enthalten sind, die Leistung der Lernmodelle verbessert. Ein weiterer modellzentrierter Ansatz enthält eine Methode zur Messung von Generalisierung in Modellen, die auf irreführenden *Weak Supervision* Daten trainiert wurden. Die Ergebnisse zeigen, dass das *adversarial Network* in der Lage ist, den Grad der Generalisierung zu kontrollieren. Ein Vergleich verschiedener State-of-the-Art-Modelle für *Weak Supervision* zeigt, dass Generalisierung und Modellleistung nicht direkt zusammenhängen.
Der Ansatz zur Wissensüberwachung für die Informationsextraktion in der Industrie etabliert eine Anleitung, wie ein regelbasiertes System aus dem Blickwinkel von *Weak Supervision* in ein maschinelles Lernmodell überführt werden kann. Neben einer Fallstudie wird der Prozess detailliert erklärt und anhand von Beispielen aus einem industriellen Informationsextraktionssystem veranschaulicht.
Der zweite Teil dieser Arbeit enthält alle Originalmanuskripte der inkludierten Artikel, nennt die entsprechenden Quellen und erläutert den Beitrag der einzelnen Autoren.

Contributing Articles

Schweter, S., & **März, L.** (2020). Triple E-Effective Ensembling of Embeddings and Language Models for NER of Historical German. In *Working Notes of CLEF 2020 - Conference and Labs of the Evaluation Forum, Thessaloniki, Greece, September 22-25, 2020.* `http://ceur-ws.org/Vol-2696/paper_173.pdf`

März, L., Schweter, S., Poerner, N., Roth, B., & Schütze, H. (2021). Data Centric Domain Adaptation for Historical Text with OCR Errors. In *International Conference on Document Analysis and Recognition* (pp. 748-761). Springer, Cham. `https://doi.org/10.1007/978-3-030-86331-9_48`

Schweter, S., **März, L.**, Schmid, K., & Çano, E. (2022). hmBERT: Historical Multilingual Language Models for Named Entity Recognition. In *Proceedings of the Working Notes of CLEF 2022 - Conference and Labs of the Evaluation Forum, Bologna, Italy, September 5th - 8th, 2022.* `http://ceur-ws.org/Vol-3180/paper-87.pdf`

März, L., Asgari, E., Braune, F., Zimmermann, F., & Roth, B. (2021). KnowMAN: Weakly Supervised Multinomial Adversarial Networks. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 9549–9557, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics. `https://aclanthology.org/2021.emnlp-main.751`

März, L., Asgari, E., Braune, F., Zimmermann, F., & Roth, B. (2022). XPASC: Measuring Generalization in Weak Supervision by Explainability and Association. Submitted to *Journal of Natural Language Engineering* on 06 May 2022, accepted (with revisions) on 03 September 2022, re-submitted on 22 November 2022. *ArXiv preprint:* `https://arxiv.org/abs/2206.01444`

März, L., Altergott, C., Stephan, A., & Roth, B. (2022). Recycle your Rules: How to turn a Rule-based System into a Machine Learning Model. Submitted to *European Chapter of the Association for Computational Linguistics (EACL) 2023* on 20 October 2022.

Contents

Part I.

Synopsis

1. Introduction and Background

1.1. Outline

This work addresses knowledge supervised information extraction in natural language processing and utilizes machine learning to learn different downstream tasks and is divided in two main parts.
Part I, the synopsis of this thesis, includes theoretical background relevant to the topic as well as a brief description of the contributing articles. After a motivation on the topic, relevant concepts related to the contributing articles contained in this work are presented and explained. First of all, the general basics of machine learning and associated algorithms are explained. After an introduction to supervised machine learning and knowledge integration, various ways of representing language data are presented. The introductory chapter ends with a presentation of relevant downstream tasks. When explaining the basics, care was taken to relate the approaches to other current research activities. Next, all contributing articles in this work are related to the appropriate area of knowledge supervised machine learning as well as roughly motivated and outlined. Starting with methods that focus on data-centric knowledge supervision, the chapter continues with model-centric knowledge supervision before finally dealing with knowledge supervision for industrial information extraction. The synopsis ends with a conclusion and an outlook on possible starting points for further work in the research area of knowledge supervised machine learning.
Finally, Part II contains the original manuscripts of all contributing articles. Each article is preceded by a description of the author's contributions and relevant meta information, such as the status of the manuscript.

1.2. Motivation

The area of artificial intelligence (AI) or, to put it another way, the possibility of enabling computers to learn independently, has been one of the pioneering research topics in recent years. In contrast to other disciplines (e.g., mathematics), the computational processing of language has a comparatively short history. Due to digital communication, digital language content is available in extremely large quantities, unlike in the past. The desire to quickly access, process and use this wealth of knowledge is obvious. Automated information extraction is an important field in this context because the extremely large amounts of data can be made accessible to humans more quickly and easily. Thus, the application of AI in natural language processing and information extraction opens up promising and relevant research areas.

Neural networks were first proposed in the 1950s, but due to a lack of computational capabilities and resources, the field almost fell into oblivion. Since their resurgence in the early 2000s, the intensity and interest of research fluctuated for a decade until neural networks finally became the focus of AI research in the 2010s.

One of the original ideas for designing neural networks was to emulate the workings of the human brain. The human brain is capable of constructing an almost all-encompassing image of reality and thus of generalizing and abstracting, while at the same time being able to acquire and access specialized knowledge on a large scale. Until today, it is a major challenge for neural architectures to mimic these human abilities.

To understand how knowledge might be integrated into neural models, one can imagine an analogy between a neural network and a drilling machine: Depending on the specific task (e.g., drilling a hole in a wooden board/ recognizing named entities), there is an optimal device (wood drill/ named entity recognition module) and a specific configuration (rotational speed/ data domain). Equipping the drilling machine with both, the correct device and the correct configuration will enable the machine to fulfill the task, i.e., to drill a hole in the wooden board. Many neural networks aim to learn various tasks from different domains without modifications, what corresponds to developing *one* drilling machine with a standard drill for any material. In contrast, the integration of knowledge into neural networks or machine learning models corresponds to developing special devices and configurations to learn different tasks. One can either develop new configurations (data-centric knowledge supervision), create new devices (model-centric knowledge supervision), or both (combining data- and model-centric knowledge supervision). In summary, knowledge supervision should enable machine learning models to learn specific problems in various domains or to broaden the representation of reality captured in those models.

When it comes to knowledge, language is one of the most important carriers of information. Every science expresses its findings and achievements through natural language. Of course, the transfer of scientific knowledge through language is complemented by visualizations or mathematical notations. However, language lies at the core of knowledge transmission. Expert knowledge is also usually available in the form of language, be it unstructured or already prepared in a structured form (e.g. taxonomies, tables, etc.). What could be more obvious than bringing all this knowledge encoded in language into the neural networks and leveraging natural language processing. The field is almost inexhaustible and extensive, making it particularly interesting. Especially in the last few years, the entire area, together with computer vision, has been at the forefront in the development of AI and a variety of available approaches can be utilized. Note that it is not important to simply integrate as much knowledge as possible into the machine learning models but relevant knowledge, for example about data, tasks, or methods.

For all these reasons, the articles contributing to this thesis lie in the field of knowledge supervision for machine learning models through natural language processing. More precisely, all solutions presented in this work deal with tasks representing the large class of custom information extraction problems encountered in practice. Developing new methods and extensions beyond the existing approaches while remaining as universal as possible enables a contribution to scientific progress as well as to applied research.

1.3. Machine Learning Background

Natural language processing is not a new branch of science and research. It dates back to the 1940s when machine translation was introduced and when in World War II written messages were encoded, sent and decoded (Johri et al., 2021). Later advances in the field include the proposal of the Turing test (Turing, 1950), the introduction of machine interpretable grammar (Chomsky, 1956) and many other rule-based algorithms. Probabilistic (Chen, 1996) and statistical (Manning and Schutze, 1999) approaches were brought up and are still being expanded to this day. As for other research branches like computer vision, machine learning or deep learning found its way into the field of natural language processing as well and has been heavily studied since then (Lewis and Gale, 1994; Jurafsky and Martin, 2000; Collobert and Weston, 2008). Most recent advances in natural language processing all play in the realm of machine learning and are built on top of neural networks. The methods proposed in this work build on machine learning and concepts relevant to the understanding of this work are explained in more detail below.

1.3.1. Machine Learning Types

In general, there are different types of machine learning, that one can distinguish between. All have in common that some kind of input should be modeled in order to obtain information from it. Throughout the remainder of this work, input means textual instances that are fed to a model. Organizing the field coarse grained, there is supervised learning, unsupervised learning, semi-supervised learning, and reinforcement learning.

Supervised learning maps the input to a pre-specified output space that is known in advance, i.e., mapping instances to labels or making a label prediction for an input instance. The basic requirement for supervised learning is labeled or annotated data. Details about supervised learning are discussed in the following section.

Unsupervised learning aims to model the input without further information or known outcome. The output of the model is not specified in advance and the most common scenario in unsupervised learning is clustering. Clustering algorithms aim to group the data into buckets that share properties in order to provide insights.

Semi-supervised learning aims to combine both, supervised and unsupervised learning to train a model and takes labeled as well as unlabeled data points as input in order to learn a downstream task.

Reinforcement learning employs two modules: a learner system and an external trainer. The trainer provides the learner system with information on how well the task has been solved. However, the learner system is not guided but must discover which action gains the most reward itself.

It is beyond the scope of this work to explain all types of machine learning in detail and there is comprehensive literature about those topics, e.g., Caruana and Niculescu-Mizil (2006), Kotsiantis (2007) or Nasteski (2017).

1.3.2. Machine Learning Algorithms

Within supervised machine learning, there are various implementations and algorithms, that can be chosen for model training. The most well-known systems include: Naive Bayes, k-Nearest Neighbours, Support Vector Machines, Decision Trees and Neural Networks.

Naive Bayes models ground on the Bayes Theorem: $P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)}$, which is used to calculate the conditional probability of A given B. Naive Bayes algorithms assume the simplification that each feature is equal and independent of another. In the machine learning context, the formula can be reformulated to determine a class y with given features X as $P(y|X) = \frac{P(X|y) \cdot P(y)}{P(X)}$. Published approaches include, among many others, an empirical study of Naive Bayes Classifiers (Watson, 2001), sentiment analysis using Naive Bayes classification (Troussas et al., 2013) or classification of brain strokes (Jayachitra and Prasanth, 2021).

k-Nearest Neighbours (kNN) assumes that instances in a data set exist close to each other in the feature space if they have similar properties. The label for an unlabeled instance can be determined by observing its nearest neighbours that already contain a label (k specifies how many neighbours should be taken into account). The original approach was proposed by Cover and Hart (1967), more recent approaches include clustering with kNN (Gallego et al., 2018) and classifying heterogeneous data sets (Ali et al., 2019b).

Support Vector Machines (SVMs) aim to find a hyperplane in a multidimensional space in order to separate/ classify data points in the space. A hyperplane is a decision boundary, with its dimensions depending on the number of features. In addition, the margin between the hyperplane and the data points should be maximized in order to separate them optimally. Maximizing the margin distance enables future data points to be classified more reliably. After the proposal of SVMs (Boser et al., 1992) various approaches were published, recent ones are aimed at the medical (Ali et al., 2019a; Ragab et al., 2019) and the bioinformatics (Liu et al., 2020) fields.

Decision Trees classify instances by sorting them based on feature values, e.g. Quinlan (1993). Each node in the tree represents a feature of an instance to be classified, and each branch represents a value that the node can accept. Instances are then classified in descending order, starting at the root node and sorted according to their feature values. During training, the nodes and branches of the tree are determined, and during testing, the optimal path for the instance is found and returned. Recent approaches using decision trees tackle Covid19 classification (Yoo et al., 2020), or turn neural networks into decision trees (Frosst and Hinton, 2017).

Neural Networks were originally intended to emulate the workings of the human brain. Basically, a neural network is a connected graph of neurons and edges that link the neurons. The neurons can either be input (having no predecessor), output (having no successor), or intermediate (having a predecessor and a successor). Each connection between neurons is responsible for transferring the output of one neuron to the input of

another and gets assigned a weight. The output of a neuron can be formalized as follows:

$$z = \sum_{i=1}^{n} x_i \times w_i \tag{1.1}$$

where x_i is the i-th feature of the input and w_i the weight for this feature. Usually, the bias term b is also added to the equation:

$$z = \left(\sum_{i=1}^{n} x_i \times w_i \right) + b \tag{1.2}$$

which is responsible for shifting the activation function towards a desired direction for successful learning. The neuron above can only perform linear operations and if that were valid for all neurons the neural network would be restricted to learning only linear input-output mappings. To also be capable of non-linear transformations an activation function is introduced:

$$z = \alpha(w \times x + b) \tag{1.3}$$

where α is the activation function and the rest of the term is a simplified description of Formula 1.2. Popular activation functions include:

- **sigmoid function** $sig(x) = 1/\left(1 + exp(-x)\right)$,
- **softmax function** $sm(x) = \Big(exp(x_i) / \Big(\sum_j exp(x_j) \Big) \Big)$,
- **hyperbolic tangent function** $tanh(x) = \left(exp(x) - exp(-z)\right) / \left(exp(x) + exp(-x)\right)$, or
- **rectified linear unit function** $ReLU(x) = max(0, x)$.

In addition, backpropagation algorithms (as used in this work) require calculating the error of the predictions and the ground truth. A Loss function is leveraged to calculate the error:

$$\text{Loss} = J(y, y_p) \tag{1.4}$$

where y is the ground truth (true label) and y_p the prediction (predicted label). Depending on the label space one has to use a regression loss function or a classification loss function. The most popular loss functions are **mean absolute error**, where the sum of absolute differences between y and y_p is calculated, **mean squared error**, where the average of the squared differences is taken, **negative log likelihood**, where the model is rewarded for making the correct label prediction with a high probability, and **cross entropy**, where the model is penalized for predicting with high confidence but wrong. For classification with discrete labels, negative log-likelihood and cross-entropy are calculated equally. See Table 1.1 for the formulae of the mentioned loss functions.
To update their weights, neural networks need a learning rule during training. The most common option is to update the weights using gradient descent (Curry, 1944):

loss function	formula
mean absolute error	$loss(x, y) = \|x - y\|$
mean squared error	$loss(x, y) = (x - y)^2$
negative log likelihood (cross entropy)	$loss(x, y) = -logP(x = y)$

Table 1.1.: Overview of loss functions. x denotes the input, y is the label.

$$*W_x = W_x - lr\left(\frac{\partial J}{\partial W_x}\right) \tag{1.5}$$

where W_x is the current weight, lr the learning rule and $\frac{\partial J}{\partial W_x}$ the derivative of the loss with respect to the weight.
In other words, the activation function computes the input of a neuron from the outputs of its predecessor neurons. The learning rule modifies the weights of the connections in order to produce a preferred output for a given input. This way inputs are processed within the network, utilizing the activation function and the learning rule to produce outputs.
There are many types of neural networks (see Schmidhuber (2015)) for an extensive overview) and only a few should be mentioned in the context of this work. Figure 1.1 illustrates the structure of the presented networks.
Multilayer Perceptrons (MLPs) were first introduced by Rosenblatt (1961) and contain at least one layer of neurons. The input layer is followed by n hidden layers that provide different levels of abstraction. The output layer serves as the prediction layer. MLPs can be used for either classification or regression tasks.
Recurrent Neural Networks (RNNs) (Rumelhart et al., 1986) can be considered an extension of MLPs, adding more connections between the neurons to process sequential data. RNNs work on the principle of saving the output of a given layer and feeding it back to the input to predict the layer's output. Instead of only connecting consecutive neurons, each recurrent cell also has a loop connection to itself (see the blue cells in Figure 1.1). That means that neuron weights can also be updated by processing previous information, e.g., of time steps. A very popular kind of RNNs is the Long Short-Term Memory Network (LSTM) (Hochreiter and Schmidhuber, 1997). Instead of standard neurons as defined in Equation 1.2, LSTMs use memory blocks. Each block contains several gates that handle its state and output. The forget gate decides which parts of the information can be discarded. The input gate controls which values from the input to update the memory state with and the output gate determines (based on the input and

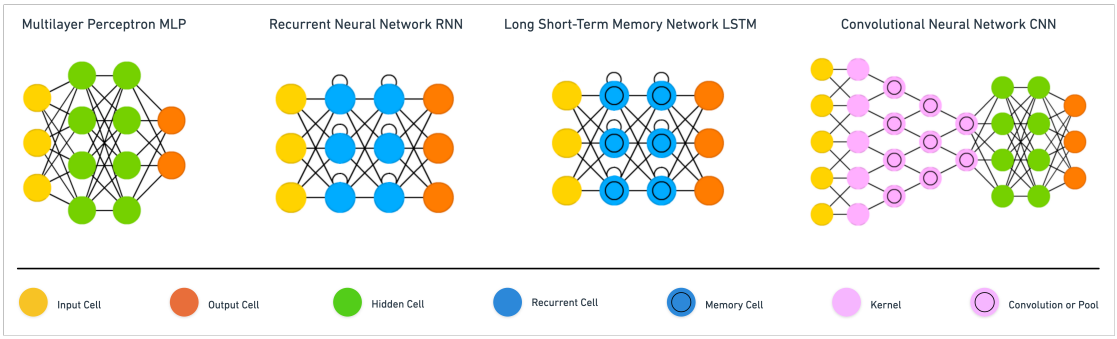


Figure 1.1.: Overview of the structure of MLP, RNN, LSTM and CNN.

the memory state) what to output. RNNs can best be used for sequence prediction tasks, including classification, regression, and generation of speech and text.
Convolutional Neural Networks (CNNs) (Fukushima and Miyake, 1982) originally had been developed for image classification. The kernel (processing input data) of a CNN is followed by three basic layers: a convolutional layer (preserving the spatial relationship between pixels in an image by learning feature maps using smaller windows of the input data), a pooling layer (reducing the dimensionality of each feature map while preserving the most important information) and a feed forward layer (usually consisting of a MLP). CNNs are able to process image data well and can learn classification and regression tasks.

1.3.3. Supervised Machine Learning

This section explains supervised learning with neural networks using backpropagation and outlines the relevant basic principles. See also Figure 1.2 for an illustration of the process.
The process summarized in a short overview: Given are instances and ground truth labels as model input and label predictions as model output. Instances and labels are transformed into vectors and fed to the model. The model learns and makes label predictions which are compared to the ground truth labels. The information about the distance between prediction and ground truth is sent back to the model (the loss). This is iteratively repeated and the final trained model is saved and in turn used to get predictions for test instances without labels.
The process explained in detail: Each supervised neural model takes instances and labels for the instances as input. Let $X = (x_1, ..., x_n)$ be the collection of input instances and $Y = (y_1, ..., y_n)$ the ground truth labels. The set of labels $C, y_i \in C$ must be specified in advance and can either lie in a continuous space (for regression) or consist of discrete values (for classification). The next fundamental step is to transform the input instances into feature vectors. For natural language processing there are different techniques to do so. Among the most common are representing each word in the input instance as its index in a pre-defined vocabulary, representing the input using tf-idf (term frequency-inverse

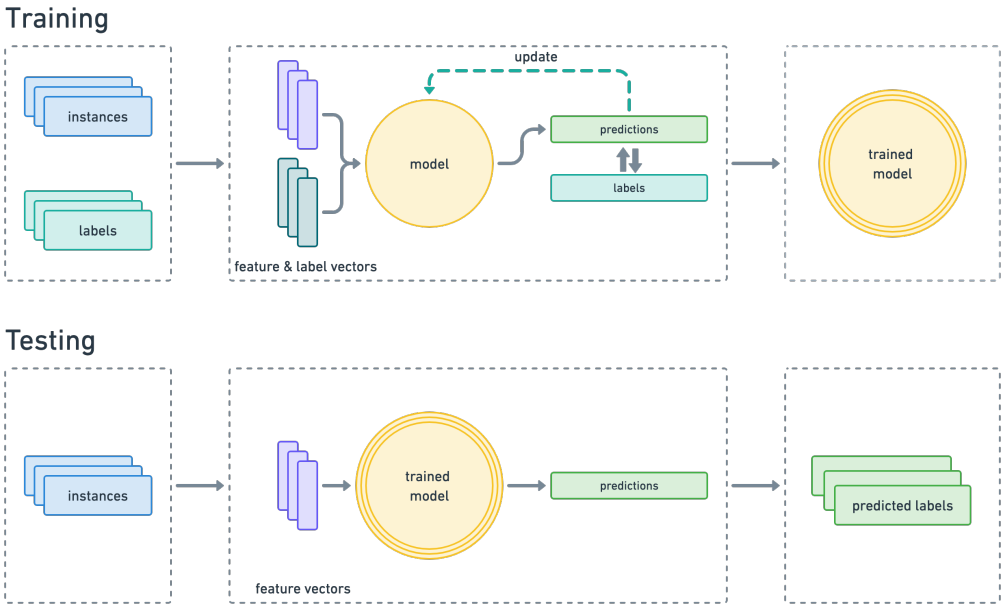


Figure 1.2.: The process of supervised machine learning and testing. Instances and labels are transformed into vectors and passed as input to the model. The model learns the inductive bias from the input instances and makes predictions for each instance. The predictions are compared to the real labels and the prediction error is calculated. This error is used to update the model. This is repeated until the model is fully trained or the maximum number of iterations is reached. During testing, only instances (no labels) are transformed into feature vectors and passed to the trained model. Based on the patterns seen in the training data the model predicts labels for the input instances.

document frequency) features or embedding each word (or subword unit) of the instance in a pre-defined feature space. Section 1.5 will discuss how to represent the input instances in more detail.
The labels need to be encoded in a vector as well and usually each label is assigned its index from the set of possible labels. Both, the resulting feature vectors and the label vectors are passed to the model afterwards.
The model detects patterns, structures, regularities, and correlations in the input instances and predicts the most likely label for each instance. Basically, the model approximates some function to map inputs to outputs:

$$y \approx f^*(x) \tag{1.6}$$

where y is the label, f^* the approximation function and x the input instance.
Neural models define a parameter mapping to learn the best approximation function:

$$y \approx f(x;\theta) \tag{1.7}$$

where y is the label, x the input instance and θ are the learnable parameters.
During the learning process, a model predicts labels $\hat{y}$ for each instance x. The label predictions $\hat{y}$ then are compared to the original known labels y, using different loss functions. Predictions are usually given in the form of probability distributions over all possible labels, which will be passed to the loss function. The calculated error or loss value is then backpropagated to the model in order to let it adjust its weights and produce more accurate predictions. The prediction, comparison to the ground truth and updating are iteratively repeated until certain criteria are met (depending on the configuration of model and training). Finally, the trained model or its state after training is saved. For testing, the trained model is loaded and test instances, again transformed to their feature vector representation, are passed to it. Based on the properties learned during training, the model predicts the most likely labels for the input instances, the probabilities are converted into labels, which then serve as model output.

1.4. Knowledge Supervised Machine Learning

The success of neural networks for solving a wide variety of tasks is resounding because they have the ability to successfully learn even complex problems that involve a lot of data. One disadvantage that neural models share across all architectures is that due to the goal of learning structures that are as general as possible, there might be a wrong specialization. Nevertheless, specialization is essential for some domains, since the domain properties can be blatantly different from, or even go beyond, general properties. Technically, the learning process of each model is predefined by its architecture and introduces an inductive bias. The inductive bias expresses all assumptions of the approximation function that the model should learn to capture the relationship between input and output. If these assumptions are no longer correct because the relationship between input and output is strongly deviating in a specialized domain or for a specific task, the model suffers from an insufficient inductive bias and the approximation function can no longer make correct predictions. However, knowledge about certain domains, tasks, and methods is often abundant, and it is highly desirable to integrate this knowledge into neural models to overcome the problem of an insufficient inductive bias. *How* to enrich neural models with additional knowledge is an interesting area of research. There are not only different approaches to tackling this problem, but also different perspectives on what "knowledge" is and how it can best be introduced into models. This work considers knowledge supervision from the following points of view: i) incorporating knowledge into neural models via the data they use (data-centric knowledge supervision), ii) integrating knowledge by focusing on the model architecture (model-centric knowledge supervision), and iii) the intersection of data- and model-centric knowledge supervision or holistic knowledge supervision. See also Figure 1.3 for an illustration of the above mentioned perspectives and some popular approaches that lie in the respective research areas. The figure also contains the section numbers of the contributing articles for each perspective. The three above mentioned perspectives and associated approaches are briefly outlined in the following, whereas the relation to this work will be highlighted in Chapter 2 in detail.

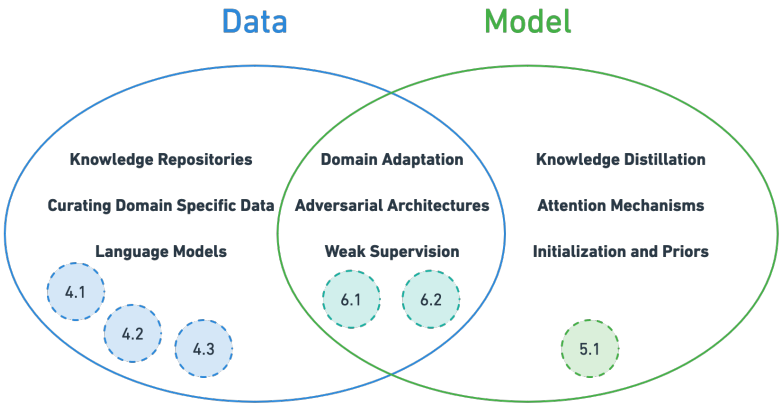


Figure 1.3.: Fields and associated topics of knowledge supervision

1.4.1. Data-Centric Methods

Data-centric approaches aim to reflect additional knowledge about the task or the domain on the data or the input the models are trained on. One could categorize integrating knowledge via data as i) using knowledge repositories, ii) curating domain specific data and iii) fine-tuning language models. Note, that this categorization is incomplete and providing a complete overview of all approaches goes beyond the scope of this work.

Using knowledge graphs as *knowledge repositories* has become very popular, especially in the last ten years. Knowledge repositories can be considered as semantic networks that model the relationship between real-world entities, such as persons, objects, events, etc. Utilizing knowledge graphs can enrich model learning with these relationships or the additional knowledge about these relationships. Ji et al. (2022) provide a comprehensive survey on knowledge graphs. In general a lot of approaches cover relation modeling problems (Nickel et al., 2016; Nathani et al., 2019) or representation learning (Xu et al., 2018; Xie et al., 2016).

Curating domain specific data means the creation and maintenance of domain specific data sets to represent the respective domain optimally. Additional knowledge about the domain to be modeled is already contained in the input data in this case or can be integrated by leveraging domain specific data in addition to training data. For natural language processing popular domain specific data sets are e.g. the SQuAD dataset for question answering (Rajpurkar et al., 2016), the Stanford Sentiment Treebank STT (Socher et al., 2013) or the IMDb movie reviews corpus (Maas et al., 2011), both for sentiment analysis.

Large *language models* will be discussed in Section 1.5 in more detail. Their pre-training or fine-tuning is another opportunity to tailor them to a specific domain and integrate additional domain knowledge. Work in this field includes the proposal of language specific models, for example, CamemBERT for French (Martin et al., 2020) or GBERT and GELECTRA for German (Chan et al., 2020), but also investigating the knowledge capturing capabilities of language models (Roberts et al., 2020) or extending language models with knowledge bases (Fei et al., 2021).

1.4.2. Model-Centric Methods

Model-centric approaches focus on the model architecture and how to design the learning process of a model in a way to exploit and incorporate knowledge optimally. While some knowledge can be contained in the input data already, additional knowledge about task and domain can be injected through other inputs or methods inside the network. For this work, we briefly outline three main principles: i) knowledge distillation, ii) attention mechanisms and iii) utilizing initialization or priors.
Knowledge distillation describes the process of reducing large neural networks in size, by still retaining the contained knowledge. No additional knowledge is brought into the model, but the goal is to compress and generalize the already present knowledge as much as possible. To perform the distillation process, knowledge is transferred from a large pre-trained model to a smaller one. The approach was first introduced by Hinton et al. (2015). Recent approaches include a reduced BERT language model called DistilBERT (Sanh et al., 2019) or propose student-teacher training (Mirzadeh et al., 2020).
In neural architectures, *attention mechanisms* are used to set the model focus on specific parts of the input that are most important for prediction. Knowledge about the most relevant input parts for a learning problem can be reflected well by using this technique. Ever since attention for neural networks was first presented and successfully applied (Bahdanau et al., 2015), it has become an integral part of the research field. Hence there are various approaches on attention mechanisms, including strategies to deal with the opposite of knowledge injection, the forgetting of learned knowledge (Serrà et al., 2018), sentiment analysis using commonsense with an attentive LSTM (Ma et al., 2018), or attentive emotion detection (Zhong et al., 2019). Extensive surveys on utilizing attention have also been published (Chaudhari et al., 2021).
The *initialization* of neural network weights or the use of *priors* can also be leveraged to achieve knowledge integration. The approaches aim to set the initialization values or use priors to adjust neural network weights in order to incorporate additional knowledge about task and domain in beforehand. Milletari et al. (2017) integrate statistical knowledge with priors to obtain robust predictions for classification with corrupted and noisy data. With informed machine learning (von Rueden et al., 2021) the authors provide a comprehensible survey on integration of knowledge via priors. Recent work proposes to perform a semantic mapping from symbolic representations of domain knowledge to weights and biases of a neural network (Hoffmann et al., 2021).

1.4.3. Holistic Methods

Approaches in this category represent the combination of the above described aspects. Relevant approaches are i) domain adaptation, ii) adversarial architectures and iii) weak supervision. The latter plays a major role in this work and is therefore described in detail in a dedicated section (see Section 1.4.4).
Domain Adaptation aims to transfer learned representations from one domain (the source domain, with large resources of labeled training data) to another (the target domain, in which there are mostly few annotated sources). The additional knowledge that is

reflected on the data (data-centric aspect) is adjusted within the model (model-centric aspect) in order to achieve good results on the target domain. Although source and target domain may differ with respect to their data properties, the same downstream task is usually learned. Though, there is also some work on utilizing domain adaptation across downstream tasks (Ramirez et al., 2019). For solving the same task, selected work includes unsupervised domain adaptation (Liang et al., 2020b), dimensionality reduction for domain adaptation (Pan et al., 2011) or knowledge aggregation techniques (Dong et al., 2021).
Adversarial Architectures have been intensively researched in recent years. Most popular are generative adversarial networks (Goodfellow et al., 2014), where two modules play a zero-sum game and one module (the generator) tries to fool the other (the discriminator). In this way, the generator is trained to produce examples that the discriminator module could not distinguish from original/real data examples. Besides, adversarial architectures are also utilized for denoising and increasing robustness of neural models (Xie et al., 2016; Warde-Farley and Bengio, 2017) or knowledge distillation (Wang et al., 2018). Adversarial models can be considered holistic, because i) additional knowledge is reflected on the instance level (data-centric aspect), since knowledge about input characteristics is utilized and ii) on the learning level (model-centric aspect), since additional knowledge about the properties of data and task are integrated here as well.

1.4.4. Weak Supervision

In general, weak supervision can be defined as a branch of machine learning where noisy, limited, or imprecise sources are used to provide a supervision signal for labeling large amounts of (training) data. With the advent of machine learning and neural networks or deep learning, the demand for large labeled data sets constantly increases. Utilizing an approach that enables the automatic labeling of large amounts of data without the need to manually perform the annotation is tempting, even if the obtained labels may contain noise.
Nevertheless, weak supervision is not the only branch of research aimed at circumventing the problem of requiring labeled training data. Before discussing weak supervision in detail, the most common approaches from which it can be distinguished are outlined:
Transductive Transfer learning (including domain adaptation, which was explained in Section 1.4.3) aims to train models on available sources of labeled data and transfer learned knowledge to another domain with less/no annotated training data (transfer on data level). There is also inductive transfer learning, i.e., the target task is different from the source task (transfer on task level). See Pan and Yang (2010) for a study on transfer learning types and paradigms.
Active learning aims to find out which data points will add the greatest value to model training. As soon as the most relevant instances have been discovered, these are labeled exclusively. Literature surveys on the field provide detailed insight, e.g. Settles (2009); Ren et al. (2022).
Also, *semi-supervised learning* can be considered a related branch, since the goal here is to leverage both, labeled and unlabeled data sources for model training in data-sparse

domains. See also Yang et al. (2021) for a recent literature survey on the topic.

Weak supervision has received increasing attention in research, especially in the last five to ten years. Early work in the area includes knowledge-based weak supervision (Hoffmann et al., 2011) and distant supervision methods (utilizing an external database for labeling) (Mintz et al., 2009). In 2017, Ratner et al. proposed *Snorkel*, a paradigm and framework to label data programmatically using weak supervision. Since then, the scientific community has been inundated with publications on weak supervision. Recent publications include a benchmark for weak supervision tasks (WRENCH) (Zhang et al., 2021), a software toolkit aiming to easily apply weak supervision in practice (skweak) (Lison et al., 2021) and a framework for interactive weak supervision (IWS) (Boecking et al., 2021), as well as a comprehensive overview on the topic of programmatic weak supervision (Zhang et al., 2022).
Within the various approaches of weak supervision, it has been agreed to use the term **"labeling functions"**. Labeling functions can be considered as templates that are applied to unlabeled data in order to match certain parts of the input and are able to assign a label once matching. Formally, assume $|X| = n$ data points and $|Y| = m$ possible labels. To retrieve weakly supervised labels for X, labeling functions $\lambda = (\lambda_1, ..., \lambda_l)$, $\lambda_i : X \to \emptyset \cup \{y\}$ are defined. Each labeling function either abstains from labeling ($\emptyset$) or assigns a class label y. Annotation by labeling functions can be expressed using two matrices: i) a matrix of labeling function matches Z, where $Z \in \{0,1\}^{n \times l}$, with $Z_{i,j} = 1$ if LF j matches instance i, otherwise $Z_{i,j} = 0$. ii) a matrix encoding the link from labeling functions to labels T, where $T \in \{0,1\}^{l \times m}$, $T_{j,k} = 1$ if label k is associated with labeling function j, $T_{j,k} = 0$ otherwise. Some models process Z and T directly, whereas for other approaches "weak" labels are computed, that are stored in matrix $\hat{Y}$, where $\hat{Y} \in \{0, ..., m\}^n$, with $\hat{Y}_i$ expressing the label index of instance i.
See also Figure 1.4 for an example of two labeling functions and their matches on unlabeled input data. The example stems from relation extraction and both labeling functions express the relation "married to". Although for each sentence one of the labeling functions matches, the second sentence does not the express the "married to" relation and is labeled wrongly. From an example like this, the system could learn spurious correlations, like: *if two persons occur in a sentence that is about a Covid19 diagnose they are married.* This is a wrong conclusion and accordingly the second sentence introduces noise and misleading signals into the model.
There are different possibilities how to generate labeling functions, for example hand-crafting them (Ratner et al., 2017; Meng et al., 2018), leveraging knowledge bases to retrieve labeling functions (Hoffmann et al., 2021; Liang et al., 2020a; Roth and Klakow, 2013), use pre-trained models or existing systems (Bach et al., 2019) or use crowd-sourced labels (Yuen et al., 2011; Nguyen et al., 2017).
To deal with the noisy and possibly misleading output of weak supervision (Z and $\hat{Y}$) there are three main groups of approaches: i) label models, ii) end models and iii) joint models.
Label models aim to denoise the labels contained in $\hat{Y}$ before passing them to a machine

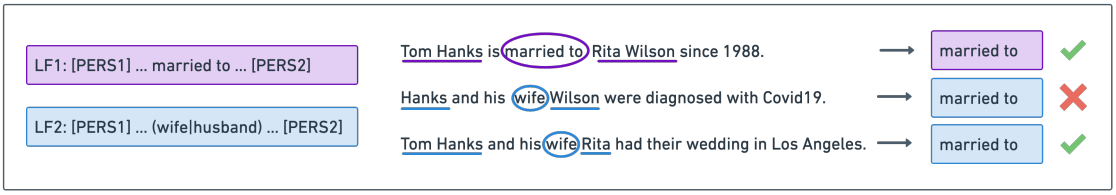


Figure 1.4.: Labeling function application to example sentences from the relation extraction task. Labeling function 1 matches the first sentence and assigns label "married to". Labeling function 2 matches the second and third sentence and assigns "married to" to both of them. The assignment is wrong for the second sentence, since it does not express this relation.

learning model for further processing. Usually there are overlaps and conflicts among the different labeling functions. In addition, some of the labeling functions might be of higher precision than others or there is a degree of statistical dependence among the labeling functions. Some label models are leveraged to model these dependencies between several labeling functions and output probabilistic labels (Ratner et al., 2016; Cachay et al., 2021b; Bach et al., 2017; Varma et al., 2017), that models designed for weak supervision can take as inputs in turn. There are also approaches that assume statistical independence of the labeling functions or output hard labels (no probability distribution, but one label per instance). This includes, for example, majority voting, where the output of all labeling functions is collected and if there is a winning label according to their votes it is assigned (otherwise either a random label/ the "other" class is assigned or the instance is discarded).

End models take the noisy labels (produced by a label model) as input and learn the downstream task on the weakly supervised data. In general, any neural model can be trained on the weakly supervised labels. However, there are also approaches that are specifically designed for learning from weakly supervised or noisy data. Some utilize a noise aware loss function (Ratner et al., 2017), others try to learn from the weakly labeled instances in a way to also leverage unlabeled data points (Yu et al., 2021). See Song et al. (2020) for a survey on 62 different noise-robust models.

Joint models integrate the two above mentioned approaches, take matrices T and Z as input and train label model and end model at the same time. Hence, they are also referred to as end-to-end models (taking the weakly supervised input and output the desired predictions for the end task). Recent work aims to learn the downstream model by alternately training a label model and using the predictions from the label model for training the end model and vice versa (Cachay et al., 2021a) or estimating the reliability of labeling functions, then aggregating weak labels (in order to denoise them) and feeding them to a neural classifier (Ren et al., 2020).

In this work, weak supervision is considered as combining data- and model-centric knowledge integration into neural networks. The labeling functions reflect additional knowledge about the task on the training data (data-centric) and additional inputs Z

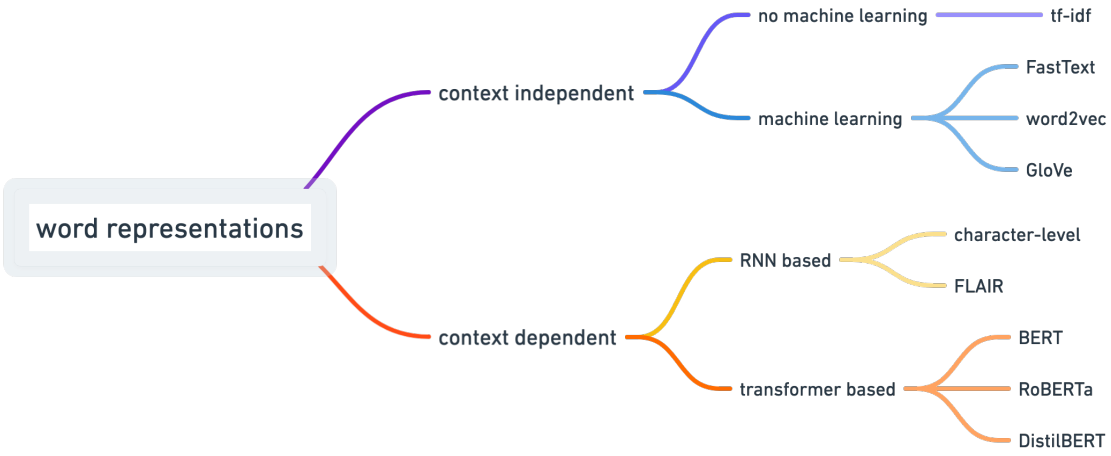


Figure 1.5.: Overview of different word representations.

and T integrate task/method knowledge on the model level (model-centric).

1.5. Representation Learning for Natural Language

In its natural form, language is encoded with letters, rather than numerically, as is usually the case with other data intended for machine processing. To enable the processing of language though, it must be converted into a machine-readable, numeric format. One of the most basic approaches to doing this is to build a vocabulary of all the words contained in a data set and represent each word in an instance as its index number in the vocabulary. For most cases, a representation like that will not be sufficient. Words cannot always be considered independent functional units, but can often only get their full meaning in the context in which they occur. There are several approaches that aim to take the context of a word into account and learn a good representation. Basically, the word representation approaches can be divided into representing words with or without taking their context into account. See Figure 1.5 for an overview of different word representations. This section provides insights and techniques for both approaches and explains the concepts relevant to this work.

1.5.1. Context Independent Representations

This section describes three different context independent word representations used in this work. While tf-idf and BPE do not require machine learning, FastText does.

TF-IDF (Term Frequency-Inverse Document Frequency) is a statistical measure evaluating how relevant a word is to a document within a collection of documents. It is calculated by multiplying the term frequency by the inverse document frequency. The term frequency expresses how often a word occurs in a document and can either simply

be counted or normalized by parameters such as word length or document length. The inverse document frequency of the word within all documents indicates how rare a word is with respect to the entire collection of documents. To calculate the idf for a word, the size of the document collection is divided by the total number of documents that do contain the word. Tf-idf can be defined as:

$$tf\ idf(t, d, D) = tf(t, d) \times idf(t, D) \tag{1.8}$$

$$tf(t, d) = log\,(1 + freq\,(t, d)) \tag{1.9}$$

$$idf(t, D) = log\left(\frac{N}{\sum_{d|t\in d, d\in D} 1}\right) \tag{1.10}$$

where t corresponds to the term or word, d is a single document and D is the collection of documents. Accordingly, a higher tf-idf for a word given a document expresses that a term/word is more relevant to the document. Compared to other representation techniques, tf-idf is less complex, which can be useful for first experiments and results or when one wants to use less computing power.

FastText (Bojanowski et al., 2017) is a very popular approach to embed words and was mainly used before transformers had been proposed. It is based on the principle of word2vec (Mikolov et al., 2013) utilizing shallow neural networks to learn word embeddings. The approach assumes that each word is composed of several character n-grams. This feature enables it to learn not only rare words but also out-of-vocabulary words. Each word is represented as a bag of its character n-grams including the whole word as well as special boundary symbols "<" and ">". Once the bag of character n-grams is computed, two different models can be used to calculate the actual embedding. The skip-gram model aims to predict a word based on its directly adjacent words. The continuous bag-of-words (CBOW) predicts the target according to a bag-of-words containing words within a pre-specified window of neighboring words.

GloVe means *global vectors* and was introduced by Pennington et al. (2014). Unlike other previously developed embedding approaches, GloVe relies on global word statistics within texts to obtain word representations. To calculate statistics, GloVe first counts word co-occurrences and stores them in matrix X, where X_{ij} denotes how often word i has occurred with word j in the corpus. Next, co-occurrence probabilities for token pairs (i, j) are computed:

$$P_{ij} = P(j|i) = X_{ij}/X_i \tag{1.11}$$

where X_{ij} is the co-occurrence count of i and j and $X_i = \sum_k X_{ik}$ is the number of times any word k appears with word i. After the statistics are calculated, GloVe performs unsupervised learning to obtain word representations. Relying on the global word statistics enables the model to learn linear substructures in the vector space. One of the contributing articles uses GloVe-based embeddings after tokenizing the corpus using Byte-

Pair-Encoding (BPE[1], Gage (1994)) to obtain word representations. This approach was suggested by Heinzerling and Strube (2018) and is referred to as *Byte-Pair-Encoding-based embeddings* in the article in Section 4.1.

1.5.2. Context Dependent Representations

In addition to the word representation techniques presented above, there are also some context dependent representations. FLAIR is a RNN-based architecture and character-based embeddings are LSTM-based, whereas BERT, RoBERTa and DistilBERT all are transformer-based. Although there are a large number of other language models (Zhang et al., 2019; Lewis and Gale, 1994; Brown et al., 2020), only language models used in this work are explained in detail.

Character-based embeddings as first introduced by Lample et al. (2016) utilize bidirectional LSTMs to process all characters in a sentence (see Figure 1.6 for an illustration). First, for all characters, random embeddings are initialized. To obtain the word embeddings, all characters of a word are passed to the LSTM to get the representation for a word. The hidden state of the forward pass and the hidden state of the backward pass are then concatenated to form the final embedding. Having trained the character embeddings, word vectors can be formed even for words that did not occur during training (out-of-vocabulary words). Note that training of the LSTM and resulting character representations are task specific and have been found useful for morphologically rich languages.
FLAIR embeddings (Akbik et al., 2018) are also called contextualized string embeddings. They can be pre-trained on large unlabeled corpora, capture a word's meaning depending on the context and model words as character sequences, which makes the embeddings more robust to misspellings. In addition, they are capable of modeling subword structures (e.g. prefixes or suffixes). Like the character-based embeddings, they use a forward-backward RNN to retrieve sequence representations from the hidden states of both directions. In addition to the forward and backward hidden states, the following hidden character states are added for each word as well: i) the output hidden state after the last character in the word, ii) the output hidden state before the word's first character. See also Figure 1.6 for an illustration of the computing of FLAIR embeddings. The former captures the semantic-syntactic information of the sentence up to this point, whereas the latter captures the semantic-syntactic information from the end of the sentence to the character. Finally, both hidden states are concatenated to represent the final embedding, containing information about the word and its context.
BERT (bidirectional encoder representations from transformers) was proposed in 2018 by Devlin et al. and has since then been heavily researched, extended and adapted.

[1]For tokenization words are decomposed into sub-tokens applying the following steps: i) the vocabulary is initialized, ii) each word in the corpus is represented as a combination of its characters and a special token indicating the end of a word ("/w"), iii) character pairs are iteratively counted within the tokens in the vocabulary, iv) the occurrence of the most frequent pair is merged and the resulting new n-gram is added to the vocabulary, v) step iv is repeated until certain criteria are met (number of merge operations, vocabulary size etc.).

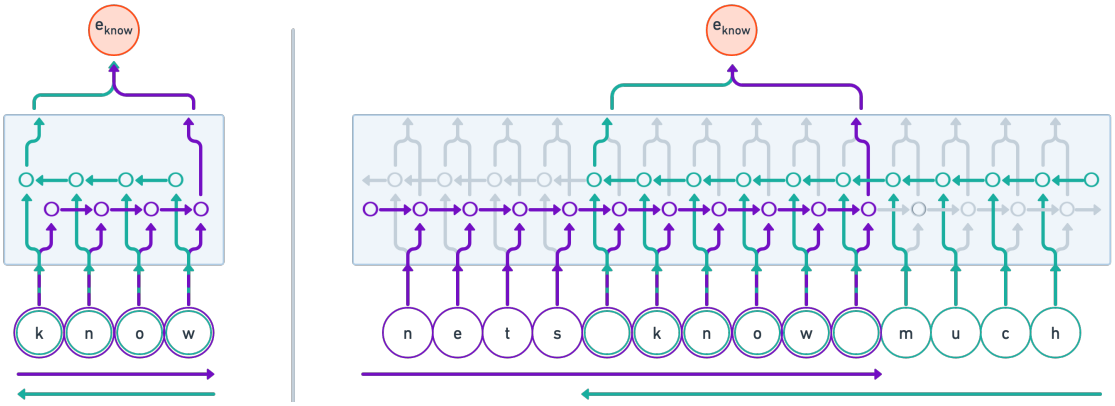


Figure 1.6.: Character-level embeddings as proposed by Lample et al. Lample et al. (2016) on the left and FLAIR embeddings as proposed by Akbik et al. (2018) on the right.

BERT is the first language model that can take into account both, the left and the right context of a word in a sentence and upon its publication it set the state-of-the-art for different important natural language processing benchmarks (e.g. GLUE or SQuAD). The core concept contains self-supervised training of 24 transformer blocks and uses masked language modeling and next sentence prediction in order to model context dependencies optimally. Masked language modeling (MLM) in this case means masking 15% of the tokens during pre-training and aiming to predict the masked words. For next sentence prediction (NSP) a simple binary classification task is designed that aims to predict whether sentence B immediately follows sentence A. By doing so, the relationship between sentences should be modeled. After pre-training, the model can be fine-tuned on various downstream tasks. See Figure 1.7 for a depiction of the BERT architecture.

RoBERTa (Liu et al., 2019) can be understood as a powerfully re-trained version of BERT where hyperparameters have been tuned more carefully in a way to increase the potential of BERT and outperform the original model. See also Table 1.2 for a comparison of BERT, RoBERTa and DistilBERT. The training data for RoBERTa is ten times more than for BERT, resulting in longer training time, but better results on the end tasks. Furthermore, RoBERTa was trained with more training iterations and the NSP was removed from the pre-training process. Also, they use a larger vocabulary and replace the original MLM by a dynamic masking process.

DistilBERT (Sanh et al., 2019) is intended to speed up the training process of BERT and to reduce the large size of BERT while retaining the performance as good as possible. In summary, DistilBERT reduces the number of parameters of BERT base by 40%, enhances the speed by 60% while retaining 97% of its capabilities. Instead of using 12/24 transformer blocks, DistilBERT uses six blocks only, and in addition, the token-type embeddings and the pooler are removed. Since larger batch-sizes and dynamic masking have been proven useful with RoBERTA, DistilBERT also uses both and removes the next sentence prediction as well. Two additional loss functions, cosine-distance loss and

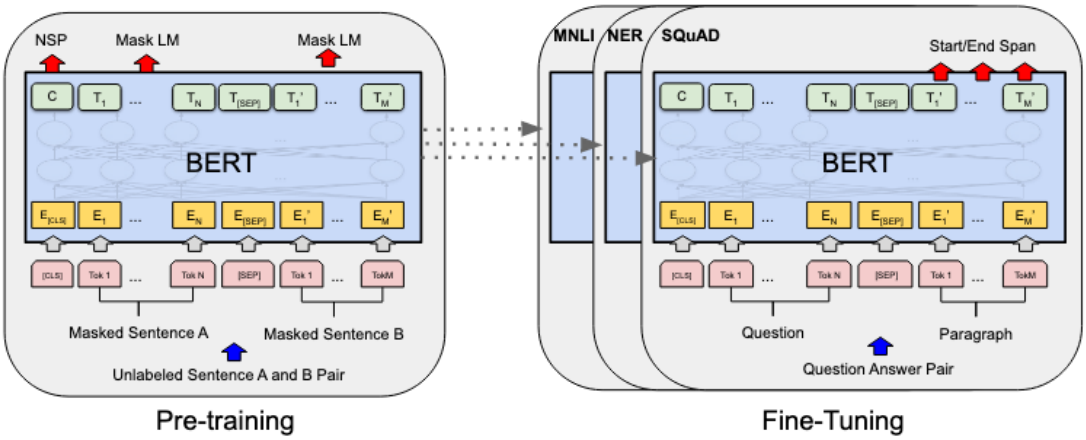


Figure 1.7.: Pre-training and fine-tuning process for BERT. Figure taken from the original paper by Devlin et al. (2019)

	BERT	**RoBERTa**	**DistilBERT**
method	BERT (= bidirectional transformer with 12 encoder blocks, MLM and NSP)	BERT without NSP, larger mini-batches, larger learning rate, bigger step size, longer training, different masking	BERT distilled (= 6 transformer blocks, no pooler, no token-type embeddings, no NSP, distillation loss, cosine-distance loss)
data	Books Corpus, Wikipedia	Books Corpus, Wikipedia, CCNews, OpenWebText, Stories	Books Corpus, Wikipedia
data size	16 GB	160 GB	16 GB
number of parameters	Base: 110 M Large: 340 M	Base: 110 M Large: 340 M	Base: 66 M -
training time	Base: 96 days Large: 256 days	4-5 times more than BERT	Base: 4 times less than BERT -
performance	upon publication (2018) outperforms state-of-the-art on GLUE benchmark	2-20% better than BERT	3% worse than BERT

Table 1.2.: Differences in BERT-based language models.

distillation loss, ensure that both the hidden states and the output are as similar as possible to BERT. DistilBERT is trained with student-teacher training.

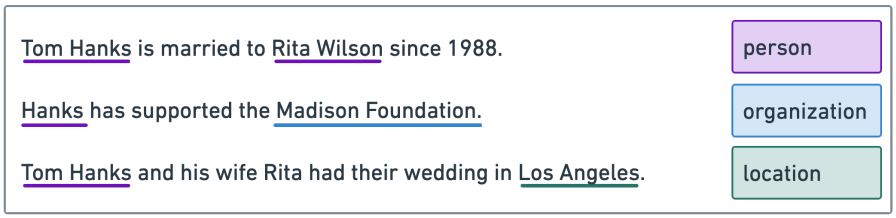


Figure 1.8.: Example sentences for Named Entity Recognition.

1.6. Downstream Tasks

In the supervised machine learning context, a downstream task means any task that utilizes a pre-trained model or component for some kind of prediction. Machine learning models are applied to a broad variety of different natural language processing downstream tasks. Among them are part-of-speech tagging, information retrieval, topic modeling, named entity recognition, relation extraction, and text classification. The latter three are also addressed within the scope of this work and are explained in detail in the next sections.

1.6.1. Named Entity Recognition

In named entity recognition, nouns and proper nouns of interest (named entities) in a text should be located and categorized. For many other tasks in natural language processing (e.g., relation extraction) it is a prerequisite to find and define the entities. Which entities are to be extracted is determined in advance and the training data must be annotated accordingly. Popular named entities that are usually extracted from sentences are **person**, **organization** and **location**. See Figure 1.8 for example sentences containing the above mentioned entities.
It is possible that any, none, or all entities occur in a sentence. Since named entities are oftentimes proper nouns, the first letter is capitalized in many languages. This pattern in the surface form is very helpful for successfully solving named entity recognition. On the other hand, there are many proper nouns in texts that should not be classified as named entities. Assume locations, persons and organizations are extracted and the input sentence is: *"Today I go to Metro to buy Lugana for my party"*. *"Lugana"* is a proper noun, but in this context neither a location, nor a person, nor an organization, but the name of a type of wine. Note that in other contexts, *"Lugana"* can be a location as well. Thus, simply finding all proper nouns and classifying them is not sufficient, and more advanced approaches are needed.
Since named entity recognition was first introduced in 1998 (Chinchor, 1998) at the Message Understanding Conference, many approaches and benchmarks have been presented to address the task. Popular benchmarks include the CoNLL 2003 data set (Sang and Meulder, 2003) for English and German or Ontonotes v5 (Weischedel, 2013) for English, Chinese and Arabic. State-of-the-art for CoNLL was established by a transformer-based

architecture in 2021 (Wang et al., 2021). Also knowledge distillation approaches (Zhou and Chen, 2021) and different derivatives from BERT had reached strong results (Schweter and Akbik, 2020; Li et al., 2020) on the above mentioned benchmarks.
Named entities can consist of several words. In Figure 1.8 most of the entities consist of two words, e.g. first and last name of a person. Various tagging schemes have been proposed to take this fact into account. Most common is the *IOB2* tagging scheme (Sang and Veenstra, 1999), where for each entity it is marked if it occurs at the beginning or inside an entity span. In the first example sentence, "Tom Hanks" would be labeled as "Tom <B-PER> Hanks <I-PER>", where "PER" is the person label, "B" marks the beginning of the entity span and "I" indicates that the word lies within the span. See also Alshammari and Alanazi (2020) for an extensive study on tagging schemes in named entity recognition.

To evaluate named entity recognition, the **F1 score** is calculated, balancing the overall precision and recall:

$$F_1 = \frac{2}{recall^{-1} + precision^{-1}} \tag{1.12}$$

where precision expresses how many instances have been labeled correctly and recall expresses how many instances of a label have been found out of all possible occurrences of the label.

$$precision = \frac{TP}{TP + FP} \tag{1.13}$$

$$recall = \frac{TP}{TP + FN} \tag{1.14}$$

where TP is the number of true positives (correct label was predicted), FP is the number of false positives (correct label was not predicted) and FN is the number of false negatives (label was wrongly predicted). Originally, the F1 score was developed for binary tasks. In order to also be able to evaluate multi-class tasks, the F1 score is either calculated per class and then the average is formed (macro F1) or TP, FP and FN are aggregated across all classes and then the F1 score is calculated from them (micro F1).
The contributing articles in this work tackle named entity recognition in the historical domain (Section 4.1 and 4.3) as well as in an industry scenario with custom named entities (Section 6.1).

1.6.2. Relation Extraction

The task of relation extraction builds upon named entity recognition because the relation between two entities should be extracted in this task. The example in Figure 1.4 addresses relation extraction, where the relation "married to" must be recognized for the first and third example sentences. Relation extraction represents a fundamental task in natural language processing, such as building knowledge graphs, sentiment analysis or question answering. Relation extraction can be conducted for binary cases (for instance, spouse vs. no spouse), but is also widely explored in a multi-class scenario, where a certain relation

from a set of relations must be predicted. In addition, a distinction must be made whether the relationship between two or between n entities must be extracted. Jiang et al. (2020) provide an analysis of this topic in their recent publication.
One challenge is that data sets for relation extraction are often unbalanced and contain a large proportion of unlabeled examples. In addition, the various relation classes in the data are also represented to different extents often, i.e., unbalanced. Another challenge in extracting relations is that there are often different words expressing the same entity that need to be disambiguated between (e.g., the entity "Rita Wilson" can be expressed as "Rita", "Mr. Wilson", "Hanks wife" etc.).
Several benchmarks for relation extraction allow to follow the recent advances in the field. The TACRED benchmark (Zhang et al., 2017) contains 41 different relation types to be extracted and is compiled of texts from the news and the web domain. The current state-of-the-art model (Wang et al., 2022) pre-trains large language models on task agnostic data sets to learn various structure prediction tasks, among them relation extraction. Another recent approach (Baek and Choi, 2022) tackles the problem of class imbalance with a minority class attention module and augmentation methods.
The SemEval 2010 data set (Hendrickx et al., 2009) focuses on multi-class classification of semantic relations between pairs of entities and contains ten different relations. The best performing model on SemEval from 2020 (Cohen et al., 2020) frames relation extraction as a span prediction problem like e.g., question answering. Another recent approach (Zhao et al., 2021) utilized graph neural networks to tackle relation extraction.

Just like named entity recognition, relation extraction is evaluated with the F1 score.
Two of the contributing articles in this work (see Section 5.1 and 5.2) address binary extraction of the *spouse* relation.

1.6.3. Text Classification

Text classification is the downstream task of assigning an appropriate category to an input sentence or text. The categories or classes depend on the data set as well as the topic and common classification problems in natural language processing include spam detection, sentiment analysis, question classification, topic classification or language detection.
Figure 1.9 shows example sentences from sentiment analysis. While the first sentence clearly expresses a positive sentiment towards Rita's dress, the second sentence expresses a negative sentiment towards Tom Hanks. The difficulties in text classification are as different as the tasks. For sentiment analysis, but also for many other tasks, a common problem is negation (of adjectives or verbs) or the occurrence of multiple adjectives in one sentence indicating different sentiments, as in the second sentence of the example. A major difficulty is that words are often ambiguous and require several sentences to unequivocally determine their meaning. Constructs such as sarcasm, irony or jokes are also difficult to handle in classification tasks. Note that the vast majority of classification models are designed for the English language, so there is a great need to transfer or adapt them to other languages.

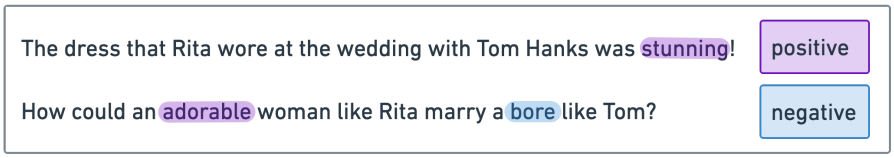


Figure 1.9.: Example sentences from sentiment classification.

Text classification benchmarks include AG News (Zhang et al., 2015) (a data set comprising news articles from four different categories), TREC (Li and Roth, 2002) (a question classification data set with six classes) or IMDb (Maas et al., 2011) (a binary movie review data set). For AGNews the best performing model is XLNet (Yang et al., 2019), a transformer based auto-regressive architecture learning bidirectional contexts by maximizing the expected likelihood over all permutations of the input sequence factorization order. The Universal Sentence Encoder (Cer et al., 2018), a popular model for learning sentence representations for transfer learning, tops the leaderboard for the TREC data set. The current state-of-the art for the IMDb data set was established in 2020 by ERNIE-doc (Ding et al., 2021) an extended language model based on BERT capable of transforming long documents.

Tasks in text classification are usually evaluated using the accuracy or the error rate. Accuracy expresses how many instances out of the total instances have been classified correctly:

$$accuracy = \frac{\#correct\ predictions}{\#total\ predictions} = \frac{TP + TN}{TP + FP + TN + FN} \tag{1.15}$$

The error rate, or simply error, is defined as:

$$error = 1 - \frac{\#correct\ predictions}{\#total\ predictions} = \frac{\#incorrect\ predictions}{\#total\ predictions} \tag{1.16}$$

Accuracy can also be expressed as $1 - error$ and error can also be expressed as $1 - accuracy$. Both, accuracy and error can be used for binary or multi-class classification.
Text classification problems are addressed by the articles presented in Sections 5.1 and 5.2.

2. Knowledge-Supervised Information Extraction for Natural Language Processing

Infusing knowledge into neural models is a challenging and complex area of research. A high proportion of knowledge is encoded in language (regardless of whether it is spoken, written, or otherwise represented). The enrichment of neural models with additional knowledge is particularly desirable in the context of natural language processing. Information extraction is only one field of natural language processing, but oftentimes it is used in customized and specialized contexts such as industry or medicine. The specialized domains have a particularly large wealth of knowledge, and information extraction systems that can access and successfully exploit this knowledge are required. This thesis addresses the following perspectives on knowledge supervision for neural models, see also Figure 2.1: Incorporate knowledge into i) the data that the models use for training, ii) the model architecture, and iii) models for industrial information extraction. The contributing articles presented in this work cover all three aspects and are explained and briefly presented in the following section. Figures 2.2, 2.3 and 2.4 illustrate in which section the original manuscript of the contributing articles for each perspective can be found.

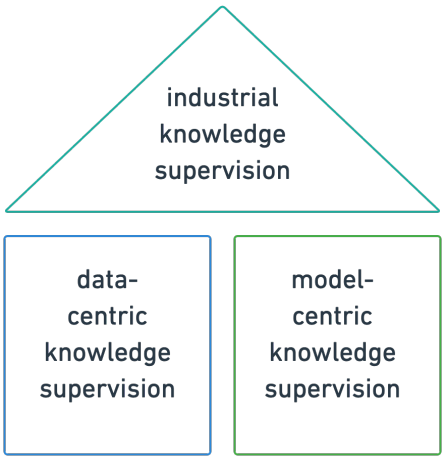


Figure 2.1.: Knowledge supervision perspectives addressed in this work.

2.1. Data-Centric Knowledge Supervision

In this part of the work, approaches to reflecting knowledge on the training data for neural models are presented. Many domains come with peculiarities that are not present outside their domain, e.g., specialized vocabulary, syntax, or, in the case that text has been digitized using optical character recognition (OCR), font type. All of these peculiarities must be captured by the inductive bias of a model and learned during training. There is a variety of approaches for model training and one might assume that using "good data" for training is sufficient and that alone should be enough for data-centricity. What is decisive, however, is that knowledge supervision integrates additional knowledge into the data. The application of supervised algorithms requires annotated data, which often cannot be assumed to be available both in terms of number of data points and representability for the domain. Furthermore, there can be a lot of valuable knowledge in unlabeled data that one wants to use without having to annotate it and there may also be further knowledge about the task or domain that should be integrated. The aim is therefore to be able to use the knowledge regardless of its source or whether it is represented by labeled or unlabeled data. Hence, the approaches presented in this section all have the same goal: optimally use, condense, and enrich existing data sources to enable efficient domain-specific training. In addition to domain specific approaches for the integration of knowledge, the articles also deal with universal methods that can be applied to any domain.

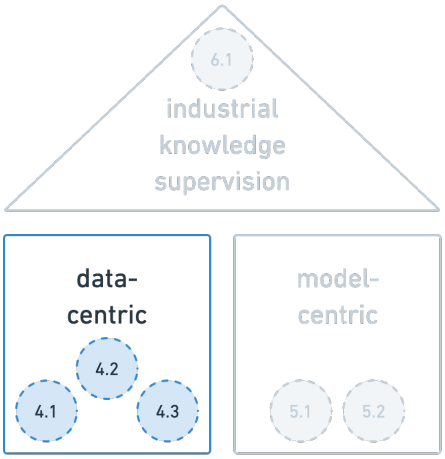


Figure 2.2.: Data-centric knowledge supervision.

2.1.1. Embedding Domain Knowledge in Input Representations.

Contributing article:
Schweter, S., & **März, L.** (2020). Triple E-Effective Ensembling of Embeddings and Language Models for NER of Historical German. In *Working Notes of CLEF 2020 - Conference and Labs of the Evaluation Forum, Thessaloniki, Greece, September 22-25, 2020.*

This approach operates in the scope of named entity recognition for the historical domain. What makes this domain different from others are several peculiarities associated with historical texts as such, but also associated with their digitization process. Historical text differs most from contemporary texts in terms of vocabulary and spelling, i.e., not only different words have been used than today, but also the phonetic shift and orthography are relevant. In addition, historical texts have not been created digitally, but were printed via letter press or even handwritten. To make them accessible for machine learning, they need to be digitized first, which is usually done by Optical Character Recognition (OCR). This process introduces errors that can either come from the OCR implementation itself or from hard-to-read documents that have suffered over time. Therefore, contemporary data sources do not provide optimal representation of the specialties of historical texts. Operating in the scope of supervised machine learning requires labeled data and for the historical domain this data is very scarce. Thus, ingesting knowledge about the historical data through a channel other than annotated data is a reasonable approach.
One data-centric way to enrich the representation of the input is to use embeddings and language models (as explained in Section 1.5). There is a large variety of embeddings for standard contemporary language (mostly collected from the news domain and web texts), which also can be leveraged for the historical domain. Nevertheless, the embeddings for standard contemporary language will not represent the historical domain well enough. For this purpose, embeddings for the historical domain can be created, and language models can either be fine-tuned or trained from scratch on unlabeled historical data. The presented article (see Section 4.1) successfully creates a "T-shape" in the knowledge representation, i.e., having a broad spectrum of knowledge covered by the large contemporary embeddings (the horizontal part of T) and at the same time going deep and covering very specific knowledge about the historical domain (the vertical part of T). This "T-shape"-principle can also be applied when operating in any other specialized domain.
To cope with coarse-grained named entity recognition for historical German texts, knowledge is integrated by assembling various embeddings and language models composed of contemporary and historical data. First, historical embeddings and language models are computed[1]. The embeddings are calculated with different embedding methods: FastText (Mikolov et al., 2018), Byte Pair Encoding-based embeddings (BPE, Heinzerling and Strube (2018)) and FLAIR (Akbik et al., 2018). The language models base on BERT (Devlin et al., 2019). For each method, different German corpora, either contemporary or historical, are used. The best performing embeddings are then determined experimentally

[1] In addition to using those for the conducted experiments, they are also publicly released.

for each category (FastText/ BPE, Flair, BERT). For the final ensemble, the best possible combination of the different embeddings and language models is found experimentally. In this way, knowledge from several sources enriches one representation in order to carry out standard model training for named entity recognition. For model training, a bidirectional LSTM with CRF as the final layer provided by the FLAIR library is used. The results show that the combination of broad knowledge with domain-specific knowledge ("T-shape"-principle) is powerful. The best performing model combines embeddings trained on contemporary (Wikipedia) texts with a language model that was trained on a large collection of unlabeled historical data. Consequently, reflecting domain knowledge on input representations is a good option in the area of data-centric knowledge supervision and it seems promising to apply this method to other domains as well.

2.1.2. Introducing Domain Knowledge using Input Perturbation

Contributing article:
März, L., Schweter, S., Poerner, N., Roth, B., & Schütze, H. (2021). Data Centric Domain Adaptation for Historical Text with OCR Errors. In *International Conference on Document Analysis and Recognition* (pp. 748-761). Springer, Cham.

This method approaches historical named entity recognition as well. Here, the goal is to extract historical named entities for French and Dutch. The focus of this article is twofold: on the one hand, domain knowledge is incorporated into embeddings and language models, on the other hand, domain knowledge is reflected by assimilating the peculiarities of historical texts via input perturbation methods. Again, unlabeled historical data is successfully leveraged to create historical language models and embeddings. In addition to the embeddings mentioned in the previous section, character-level embeddings are used. Words are embedded using the representations of their individual characters, thus it is possible to represent out-of-vocabulary words[2] better. This is a favorable advantage for the historical domain because due to the deviations from the modern language and the OCR errors, out-of-vocabulary words are common.
Although knowledge from unlabeled resources can be integrated well by using embeddings and language models, a larger amount of labeled data for training is desirable too. When a neural model sees a greater variety or number of typical examples in the input, it is more likely to learn an appropriate inductive bias and make correct predictions. The article proposes to generate labeled training data by utilizing knowledge about the historical texts and computer vision methods. Labeled modern data is used for this purpose, which is then approximated to the surface form of historical texts by means of artificial perturbations. Two perturbation methods are proposed: i) introducing systematic corruptions, by *inserting*, *removing* or *transposing* characters in the contemporary labeled data and ii) perturbing contemporary labeled data by assigning a *random font* and *font size*, *printing* it to a PDF, *rotating*, *blurring* and *dropping out pixels* in the PDF and then *OCR it again*. Both methods introduce knowledge into the data in order to recreate the characteristics

[2]Words that not have been observed during training but during testing.

of OCR'd historical data. After that, the corrupted/perturbed and clean texts are aligned to retrieve the annotations for the corrupted/perturbed text. That way the peculiarities contained in the surface forms of historical texts are introduced, while also obtaining annotations for the data.
The article performs in- and cross-domain experiments, i.e., training on contemporary data and testing on historical data vs. training and testing on historical data. The in-domain experiments reaffirm that embeddings and language models are good at reflecting knowledge about the historical domain. The cross-domain experiments show that integrating knowledge by using embeddings and language models together with perturbation methods is all the more effective the more the texts used for testing contain characteristics of the historical domain. For both scenarios and both languages, state-of-the-art results are established.

2.1.3. Utilizing Focused Language Models to Reflect Domain Knowledge

Contributing article:
Schweter, S., **März, L.**, Schmid, K., & Çano, E. (2022). hmBERT: Historical Multilingual Language Models for Named Entity Recognition. *Proceedings of the Working Notes of CLEF 2022 - Conference and Labs of the Evaluation Forum, Bologna, Italy, September 5th - to - 8th, 2022.*

The last of the data-centric methods presented in this chapter also deals with named entity recognition in the historical domain. The approach is based on different aspects: i) generating very large language models in the target (historical) domain, ii) integrating multiple languages, and iii) focusing the language model on the knowledge most relevant to the downstream task. The integration of domain-specific knowledge via embeddings and language models had proven itself in the two previous works. Therefore, this method is successfully repurposed in this approach.
Some of the data used in the article (see Section 4.3) stems from historical commentaries[3] from different languages. Commentaries usually contain additional information about the text they are addressing. Because they often quote the original literary text, several languages can be found in there. Consequently, it makes sense to leverage knowledge from multiple languages when extracting information from these commentaries. In addition, linguistic properties are shared across multiple languages (Aikhenvald et al., 2006), which is another argument to use more languages. Accordingly, multiple languages are used to create a large language model (*hmBERT*) and to incorporate all knowledge contained in the languages. German, French, Swedish, Finnish, and English data sets are used to train the *hmBERT* language model. In order to avoid redundancies and to tap synergies between the different languages, a multilingual vocabulary is created for the *hmBERT* training. In order to alleviate the problem with erroneous surface shapes caused by the

[3] A classical commentary is a scholarly publication that aims to facilitate the reading and understanding of classical works of literature by providing additional information such as translations or bibliographic references.

OCR process, text is filtered out where incorrect OCR recognition is likely to occur often (based on OCR confidence).
Training large language models takes a lot of computational resources. In order to find out whether the integration of multiple languages really offers a great advantage and whether it is worth investing these resources, monolingual models are also calculated. The difference in performance on news data is not particularly large for the majority of the languages. Accordingly, depending on the needs and available resources, either the monolingual or the multilingual models can be used. Overall, *hmBERT* proved to be successful and could outperform the baseline for three out of four languages on the NewsEye NER dataset (Hamdi et al., 2021).
To finally achieve focusing of *hmBERT* on the knowledge most relevant to the commentary task, multi-stage fine-tuning is performed. Hereby, in the first stage, the multilingual model is fine-tuned. Hyperparameter configurations for the best performing model are then used for the second stage of fine-tuning, where one optimal model is determined for each language. The final model successfully addresses the commentary challenge and can outperform other systems for two out of three languages. Hence, the results show that the fine-tuning of language models is helpful for the optimal reflection of domain knowledge.

2.2. Model-Centric Knowledge Supervision

Another way to approach knowledge supervision is to put the main focus on the model architecture instead of the input data. There are many ways to influence the learning of the appropriate inductive bias in the way the input data is processed, projected, transformed, and exploited in neural networks. Sometimes one has to operate in a scenario where there is no way to make changes, adjust, or enrich the data with knowledge. In addition, the ingestion of knowledge based on the architecture of a network can tackle other dimensions and aspects. Neural networks process the input data systematically and stringently. Therefore, they can learn and recognize structures that are not (immediately) accessible by a human. In addition, these structures can certainly differ from those that a human considers obvious. Taking this fact into account, it is desirable to design and develop architectures that allow the incorporation of knowledge based on the internal workings of the model. The article outlined in the next section falls within the scope of model-centric knowledge supervision.
Furthermore, there is another article (März et al., 2019) towards this direction which is not included in the contributing articles for this thesis but still worth mentioning. It proposes an approach for domain adaptation by integrating additional knowledge about the target domain by adjusting transferred weights from the source domain to learn part-of-speech tagging for Twitter data. The source-domain model is trained on a large German newswire corpus and the learned weights are transferred by using them as a prior for training the final model on the target domain (a data-set of German Tweets), which obtains state-of-the-art results. For detailed information see the publication:
März, L., Trautmann, D., & Roth, B. (2019). Domain adaptation for part-of-speech tagging of noisy user-generated text. In *Proceedings of the 2019 Conference of the North*

American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers).

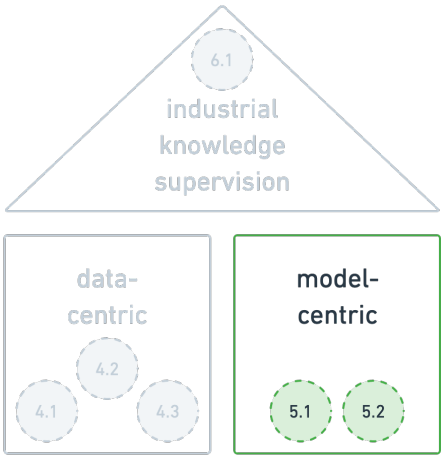


Figure 2.3.: Model-centric knowledge supervision.

2.2.1. Capturing Knowledge by Expanding the Model Focus

Contributing article:
März, L., Asgari, E., Braune, F., Zimmermann, F., & Roth, B. (2021). KnowMAN: Weakly Supervised Multinomial Adversarial Networks. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 9549–9557, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

This approach addresses one of the main difficulties resulting from weak supervision. Due to the nature of the weak supervision process, labels may contain noise or be inaccurate. Labeling functions are inflexible and will bluntly match anything they are generally designed to match, regardless of context or whether it makes sense. As a result, the instances are labeled, but they also contain noise that needs to be dealt with. In addition, during the learning process, models may over-rely on the signals captured by labeling functions and are hindered to exploit other signals. Therefore, it is desirable to get the model to focus on other relevant signals than the possibly misleading labeling function signals by integrating knowledge about weak supervision. The contributing article addresses just that — it proposes an architecture independent model, KnowMAN, with the goal of expanding the model focus to take into account all signals and not just those of the deceptive labeling functions. KnowMAN is designed in an adversarial manner, consisting of a shared feature extractor, a text classifier (learning the downstream task) and a labeling function discriminator (learning to determine which labeling function

matched for an instance). As input, it takes labeled training data as well as additional knowledge about the weak labels, captured by an input matrix reflecting which labeling function was responsible for assigning the label to the instance. The classifier and the discriminator play a statistical min-max game, i.e., each module aims to maximize its own reward while minimizing the other's reward. That translates to expanding the focus of the model on the shared feature extractor level. Both modules back-propagate to the shared feature extractor to adjust the feature representation to be optimal for solving their task. While the classifier is performing standard back-propagation to optimize downstream performance, the discriminator backpropagates the reversed gradient to optimize labeling function distinction. Reversing the discriminator gradient for backpropagation enables discarding information about the labeling functions. On the one hand, deceptive signals of the labeling functions are discarded, on the other hand, important information from other signals relevant to classification is learned. Discarding the information coming from the labeling functions entirely can be harmful, as they may still contain valuable information. Therefore, KnowMAN utilizes a hyperparameter with which the degree of discarding labeling function signals can be set. The concrete value of this parameter can then be determined experimentally, depending on the data set. The results confirm the proposed functionality, and KnowMAN improves classification results across all tested data sets.

2.2.2. Assessing Generalization of Knowledge-Supervised Systems

Contributing article:
März, L., Asgari, E., Braune, F., Zimmermann, F., & Roth, B. (2022). XPASC: Measuring Generalization in Weak Supervision by Explainability and Association.
Submitted to *Journal of Natural Language Engineering* on 06 May 2022, accepted (with minor revisions) on 03 September 2022, re-submitted on 22 November 2022.

Many approaches deal with weakly supervised or other knowledge enriched data. Usually, the success of the models is measured by their prediction performance on downstream tasks. This is reasonable, but it does not give information about the processing of the knowledge and to what extent it is utilized. Weak supervision suffers from the problem of many but noisy labels being produced. There are algorithms designed for processing weakly supervised data that can deal with the noisy labels or aim to filter them out for training. Ultimately, they all have the same goal: learning general patterns from the noisy input data. However, also in this scenario, there is no universal way to figure out if the model has generalized from the noisy data and to what extent. The proposed method in this section, XPASC, provides a novel metric to measure just that, the generalization ability of a weak supervision model, i.e., the ability to generalize from the labeling function signals. This allows gaining insights into knowledge exploitation and evaluation of models besides pure prediction performance. Different weak supervision models are evaluated with respect to this aspect, and in addition, the functionality of KnowMAN[4]

[4]Section 5.1, expanding the focus of a model to take other parts of the input besides the labeling functions into account.

is examined.
To measure the generalization from labeling function signals of a model, two aspects are taken into account: i) the explainability or importance of a feature for the model, and ii) the association of the feature to its class and its labeling function. In this way, one should be able to measure the generalization from labeling functions because, if the model associates the most important features more closely with the class, there is less risk of emphasizing the noisy signals of the potentially misleading labeling functions. To compute explainability, a method from XAI (eXplainable AI), namely *occlusion* is used. Occlusion works by comparing the model output for an instance to the model output for a modified instance. For XPASC, the Kullback-Leibler-Divergence between the prediction for an entire instance and the prediction for the instance with one omitted feature is calculated. The smaller the divergence, the less the omitted feature changes the prediction and the less important it is. That way, the explainability of the omitted feature can be determined. To compute association to classes and labeling functions, two different methods are proposed. Chi-square-based association, which relies on statistical association strength, and positive pointwise mutual information-based association, which measures the more general information-theoretic association strength. To relate class association to labeling function association, the values are subtracted for each feature. A smaller subtraction value indicates a stronger association with the labeling function, while a larger value indicates a stronger association with the label. Ultimately, both components (explainability and association) are related in the final XPASC formula and generalization of a model given a weakly supervised data set can be measured.
The proposed metric gives immediate insights into how different weak supervision models generalize from labeling function signals. On the one hand, XPASC confirms the hypothesis of KnowMAN: The hyperparameter λ in the KnowMAN architecture controls the degree of generalization. On the other hand, the generalization of different models is compared and it is shown that generalization is not the sole key to good downstream performance. The XPASC formula is flexible in that the individual components can be exchanged and other methods of explainability or association could be used and plugged in easily. In addition, the metric is novel because until now there was no universal measure to calculate the generalization from labeling functions in weak supervision settings.

2.3. Knowledge Supervision for Industrial Information Extraction

In practice there is a large class of custom information extraction problems, for example in the medical or industrial domain. In industry, there is a large collection of knowledge that is often not utilized optimally or not integrated into machine learning models at all. What makes the industrial domain challenging is that knowledge exists in many different forms, be it unstructured, structured, or in the form of expert knowledge, heuristics, databases, etc. In the automotive industry, for example, suitable learning requires valuable additional linguistic information (there is domain-specific vocabulary, words may have a different frequency and relevance to the subject than in standard texts, etc.). Basic

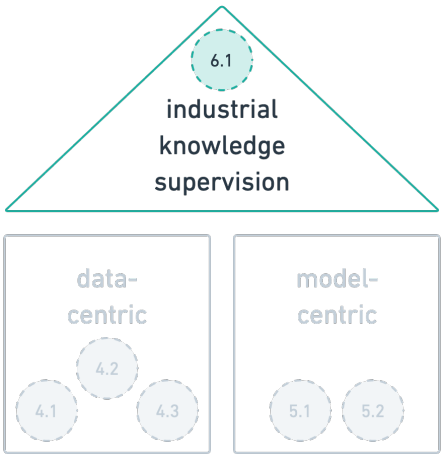


Figure 2.4.: Knowledge supervision for industrial information extraction.

meanings in this domain may also differ from other domains. For example, in sentiment analysis, the sentiment of *slow* and *fast* is different in the automotive context. In relation to food (fast food/ slow food) *slow* has a positive connotation, in relation to vehicles it is vice versa (fast car/ slow car) and *fast* has a positive connotation. However, based on the observations from data- and model-centric knowledge supervision, it is promising to utilize industrial knowledge to ingest additional information into the learning process. The data-centric side makes it possible to integrate domain-specific knowledge, e.g., in the form of valuable knowledge from domain experts. The model-centric side allows to model the knowledge enriched input in a way to optimally exploit the knowledge contained in the input as well as to integrate further knowledge. The article presented in the following section addresses knowledge supervision for industrial information extraction and takes the data- as well as the model-centric perspective into account.

2.3.1. Practical Knowledge Infusion for Industrial Information Extraction

Contributing article:
März, L., Altergott, C., Stephan, A., & Roth, B. (2022). Recycle your Rules: How to turn a Rule-based System into a Machine Learning Model.
Submitted to *European Chapter of the Association for Computational Linguistics (EACL) 2023* on 20 October 2022.

Data-centric AI and weak supervision are popular approaches to acquiring large amounts of labeled data for model training. Especially in industrial settings, there is a great demand for these methods. Usually large amounts of data are available, that reflect domain specific properties and knowledge well, but no annotations. However, what can be found in many companies are rule-based algorithms. Valuable knowledge from domain experts can be translated into rules easily and since machine learning systems

require annotated data, many companies have so far resorted to a rule-based approach. Meanwhile, however, there is a great need for companies to also use machine learning algorithms.
The contributing article addresses these issues. It utilizes a rule-based system for weak supervision in order to perform machine learning with the data obtained in this way. At the same time, a practical and theoretical guide is created, detailing the necessary steps to turn a rule-based model into a machine-learning model. First, the machine learning background for such an undertaking is explained, then the requirements for weak supervision are discussed. Usually weak supervision methods rely on two matrices, one containing the labeling function matches for each instance and the other holding the mapping from rules to labels. How to acquire both is thoroughly discussed in the article. Apart from the mathematical background, it is carefully explained how to obtain labeling functions from a rule-based system. Hereby difficulties and stumbling blocks are also discussed and how to deal with them. This part of the work covers the data-centric perspective of knowledge supervision.
After all the basic steps of the process have been expounded, a case study that addresses custom named entity recognition in an industrial setting follows. Using the case study, the individual steps of how to transfer the rule-based system into a machine learning system with the help of weak supervision are concretely illustrated and explained. Various weak supervision algorithms are then applied to the weakly supervised data. In addition, the adversarial KnowMAN architecture is adapted to also be able to learn sequence tagging problems. This part of the approach covers model-centric knowledge supervision.
The case study shows that the method is useful and worth applying in industry because the utilization of the rule-based system for weak supervision and model training improves the task's prediction performance. Overall, the high-level perspective and the concrete level can be combined with the case study and thus a decent amount of universal information can be transported. Careful attention has been paid to describing and discussing the aspects relevant to this process, so that the guide can help researchers and practitioners solve similar problems.

3. Conclusion

3.1. Research Contributions

This thesis addresses the integration of knowledge into supervised machine learning models for natural language processing. Three perspectives performing knowledge supervision are examined: data- and model-centric knowledge supervision as well as knowledge supervision for industrial information extraction. For each perspective at least one approach is proposed and a specific as well as a general contribution can be derived, which are presented below.

All articles for data-centric knowledge supervision address named entity recognition in historical documents. The specific contributions of all approaches are the training of historical taggers that establish very good results or even state-of-the-art performances for various data sets and languages (German, English, French, Dutch, Swedish, and Finnish) as well as the release of code and models. In addition, the approaches successfully outline and apply the "t-shape" principle (the combination of broad knowledge and deeply specialized knowledge when learning downstream tasks) and explain how to utilize unlabeled data for knowledge integration in general.

In model-centric knowledge supervision, the specific contribution of the first article is the development and proposal of KnowMAN, an adversarial multinomial neural network for weakly supervised datasets that is invariant to the interference of noisy weak supervision signals. The general contribution that can be drawn from this article is that a broad and robust representation of input instances is beneficial. It makes sense to review the model focus while learning representations and, if necessary, shift it from the weak signals towards other signals for improved performance on downstream tasks. The specific contribution of the second article is the proposal of a novel metric to measure generalization from misleading signals of noisy weak supervision data as well as a detailed explanation and experimental validation of the measure. Since the components of the metric are interchangeable and can therefore be tailored to different needs, it can also be used in a modified form to measure the generalization of noisy data other than weakly supervised data.

Finally, the article on knowledge supervised information extraction in industry offers a step-by-step guide on how to turn rule-based systems into machine learning models with weak supervision. This detailed guide and its application to an automotive industry task can be considered the specific contribution. In general, the recycling of existing

systems for weak supervision is another contribution to the research community and can be applied in many scenarios.

Overall, the approaches suggest that knowledge supervised information extraction is advantageous in many settings, since general representations are often insufficient to solve niche problems and there might be additional knowledge that is not covered and should be integrated in machine learning models.

3.2. Concluding Remarks

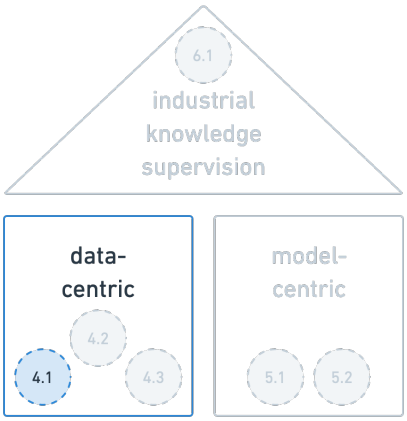
It can be expected that machine learning research and research in natural language processing will continue to advance in the upcoming years. In addition to the general improvement of model architectures and the processing of large amounts of data with increased capacities, the integration of knowledge into these models will be crucial. There are many different sources of general knowledge as well as specialist knowledge about tasks and domains. However, knowledge supervision is far from exhausted and innovative approaches are needed. This doctoral thesis aims to contribute in part to methodically equipping the research on this topic and to provide inspiration for further work in the field of knowledge supervision.
All articles in this thesis have shown that the integration of additional knowledge about domain or task is helpful for the learning process of machine learning models. The approaches provide different starting points for future work, be it i) to develop methods to better represent language-specific and cross-language knowledge, ii) to design additional inputs (such as for weak supervision) to support the learning process of the models, or iii) to develop a possibility of presenting knowledge that is already contained in models in an interpretable way for humans, in order to know which knowledge components are still missing.
Whether at some point "knowledge saturation" is reached in machine learning models or whether completely different techniques have to be developed in order to inject additional knowledge into learning problems remains an exciting open question.

Part II.

Contributing Articles

4. Data-Centric Knowledge Supervision

4.1. Triple E - Effective Ensembling of Embeddings and Language Models for NER of Historical German.



Contributing Article

Schweter, S., & **März, L.** (2020). Triple E-Effective Ensembling of Embeddings and Language Models for NER of Historical German. In *Working Notes of CLEF 2020 - Conference and Labs of the Evaluation Forum, Thessaloniki, Greece, September 22-25, 2020.*
`http://ceur-ws.org/Vol-2696/paper_173.pdf`

Copyright Information

Author Contributions

Stefan Schweter suggested taking part in the shared task and the proposed approach was discussed between both authors. Stefan Schweter took on the model training implementation, the extensive task of selecting suitable hyperparameter combinations for setting up the experiments and pre-trained most of the embeddings and language models. Luisa März made a substantial contribution with the implementation of the pre-processing of the shared task data, the downstream task and the prediction and evaluation script. She also performed the analysis of the results. Both authors supported each other through continuous revisions. The paper was mainly written by Luisa März and reworked by both authors.

Supplementary Material

Code and models: `https://github.com/stefan-it/clef-hip`

Triple E - Effective Ensembling of Embeddings and Language Models for NER of Historical German.

Stefan Schweter[1] and Luisa März[2]

[1] Bayerische Staatsbibliothek München, Digital Library/Munich Digitization Center
stefan.schweter@bsb-muenchen.de
[2] Center for Information and Language Processing (CIS), LMU Munich
maerz@cis.lmu.de

Abstract. Named entity recognition (NER) for historical texts is a challenging task compared to NER for contemporary texts. Historical texts come with several peculiarities that differ greatly from modern texts and large labeled corpora for training a neural tagger are hardly available. In this work we tackle NER for historical German with an ensembling approach, combining different labeled and unlabeled resources of historical and contemporary texts as part of the CLEF HIPE 2020 evaluation lab. We stack different word/subword embeddings and transformer-based language models to train a powerful NER tagger for historical German. We conduct experiments with different word embeddings, Flair embeddings and pretrained Bert models. The named entities are classified in literal and in metonymic sense, for which we have developed a separate tagger each. Our experiments show that the usage of Bert is particularly helpful, when trained on a large amount of historical data. Our best ensemble is a combination of FastText embeddings trained on German Wikipedia, Flair embeddings trained on CLEF HIPE data (historical German) and a Bert language model trained on a large corpus of historical German. We release our code and models[3].

Keywords: Named Entity Recognition · Transformer-based language models · Embeddings · Historical texts · Flair · FastText · Byte Pair Encoding.

1 Introduction

In NER neural networks achieve good accuracy on high resource domains such as modern news text or Twitter ([2, 4]). But on historical text, NER taggers often perform poorly. This is due to domain shift and to the fact that historical texts contain systematic errors not found in modern text, since historical datasets

[3] Our code and models are available at: https://github.com/stefan-it/clef-hipe

usually stem from optical character recognition (OCR). OCR is noisy and the Gothic type face (Fraktur) is a low resource font, that is very challenging for OCR. Another problem is that a large amount of data is required when training neural models and only relatively small corpora (e.g. [20]) exist for historical NER. All of these challenges mean that NER for contemporary texts differs greatly from NER for historical texts and that existing models cannot be used. From a resource orientated and ecological point of view it is reasonable to reuse existing models to save both computing power and emissions. Therefore, we reuse existing models on the one hand and make our newly developed language models publicly available on the other hand.

In the NLP community there are several approaches and models provided, one of which is Flair [1]. Flair allows to apply state-of-the-art natural language processing (NLP) models, such as NER, part-of-speech tagging (PoS), word sense disambiguation or classification to various input texts. In this work we built our systems with that framework.

Transformer-based language models are widely used and Bert [8] can be considered as a powerful standard resource. There are several recent approaches that use Bert for NER in different languages, such as [25] or [16]. The latter conduct experiments with historical German using Bert and unsupervised pretraining on a large corpus of historical German texts together with supervised pretraining on a contemporary German corpus.

1.1 Task and Objective

In this work, we address neural NER tagging on historical German data. With our approach we aim to solve coarse grained NER in the CLEF HIPE shared task [11] (bundle 4) for historical German as best as possible. The tagset of the provided data contains person, location, organisation, product and time. The organizers arranged two scenarios to be solved: NER for the literal sense of the words and NER for metonymic sense. The example below shows that the tags for the literal (first sentence) and metonymic (second sentence) sense can differ. *Hannover* can be interpreted as an organization as well as a location depending on its context and the metonymic category addresses this issue.

Example:
Unterhandlungen über das Konkordat mit ***B-loc Hannover*** *schreiten voran.*
Unterhandlungen über das Konkordat mit ***B-org Hannover*** *schreiten voran.*
(Negotiations on the Concordat with Hanover are progressing.)

This paper is structured as follows: The next section describes data sets and other resources that are used in the experiments presented. Section 3 outlines our method and section 4 explains details on implementation and the conducted experiments. The outcome of the experiments is discussed in that section as well. Then, section 5 overviews ideas for future work and we conclude the paper with section 6.

2 Data and Resources

This section describes the data provided by the shared task organizers as well as additional resources and data that we used for our experiments.

2.1 CLEF HIPE Data

The shared task corpus for German is composed of articles sampled among several Swiss and Luxembourgish historical newspapers on a diachronic basis and is provided by the CLEF-HIPE-2020 organizers. The articles that were chosen for the train, development and test data are journalistic articles only, that had to match certain selection criteria such as length or format. Feuilleton, tabular data, crosswords, weather forecasts, time schedules and obituaries were excluded as well as articles that were fully illegible due to massive ORC noise. The newspaper content stems for the time period from 1798 until 2018 and thus there is different OCR quality present in the data which covers a broad spectrum of text composition. The corpora were manually annotated by native speakers according the HIPE impresso guidelines ([10, 9]).

2.2 Additional Data and Resources

Table 1 gives an overview of all resources and shows time period and domain of each data set. The sizes of the training data used for the embeddings/models is shown in Table 2. Our approach includes data from different time periods as well as from various domains to reuse existing resources optimally.

Embeddings We use different **FastText**-based word embeddings [19] trained on Wikipedia[4], Common Crawl[5] and on historic data (provided by the organizers) as well as **Byte Pair Encoding**-based embeddings (BPE, [24]) trained on Wikipedia. We use the FastText embeddings trained on Wikipedia (*FastText Wiki*) and Common Crawl (*FastText CC*) in a "classic" word embeddings manner, that means we do not use subwords. To include subword information we use German subword embeddings [12] with a dimension of 300 and a vocab size of 200k (*BPEmb*). Additionally, we experiment with multilingual subword embeddings [13] with a dimension size of 300 and a vocab size of 1M (*MultiBPEmb*).

We use **Flair** embeddings [3, 2] provided by the organizers (*CLEF-HIPE*) and compared them to other Flair embeddings that were trained on historic data. We use two historic Flair embeddings that were trained by [23]: embeddings trained on the *Hamburger Anzeiger* newspaper corpus (*HHA*) and embeddings trained on the *Wiener Zeitung* newspaper corpus (*WZ*). Both embeddings are available in the Flair framework. In addition we use the data of the recently published REDEWIEDERGABE corpus [6] that consists of fictional and non-fictional texts. We also experiment with the Flair embeddings provided by [3] (*German* Flair).

[4] `https://fasttext.cc/docs/en/pretrained-vectors.html`
[5] `https://fasttext.cc/docs/en/crawl-vectors.html`

usage	name	time period	domain
train data		1798 - 2018	news
FastText	FastText Wiki	contemp.	various
FastText	FastText CC	contemp.	various
BPE	BPEmb	1798 - 2018	news
BPE	MultiBPEmb	contemp.	news
Flair	HHA	1888 - 1945	news
Flair	WZ	1703 - 1875	news
Flair	Redewiedergabe	1840 - 1920	various
Flair	*German* Flair	contemp.	various
Flair	CLEF-HIPE	1798 - 2018	news
Bert	*Europeana* Bert	1618 - 1990	news
Bert	*German* Bert	historical	various

Table 1. Overview of time periods and domains of the training data used for the embeddings and language models.

usage	name	data	tokens	size
train data		CLEF HIPE*	0.071	S
FastText	FastText Wiki	Wikipedia	1400	L
BPE	BPEmb	Wikipedia	$\approx$ 1400	L
BPE	MultiBPEmb	Wikipedia	$<$ 7000	L
FastText	FastText CC	Common Crawl	65648	XL
Flair	Redewiedergabe	REDEWIEDERGABE	0.489	S
Flair	*German* Flair	OPUS project	500	M
Flair	HHA	Hamburger Anzeiger	742	M
Flair	WZ	Wiener Zeitung	802	M
Flair	CLEF-HIPE	CLEF-HIPE*	1722	L
Bert	*Europeana* Bert	Europeana	8000	L
Bert	*German* Bert	-	$\approx$ 24000	XL

Table 2. Overview of different training data used. Number of tokens is given in millions. * indicates that data was provided by the organizers.

Transformer-based language models For transformer-based language models we conduct experiments with self-trained BERT models, *Europeana* BERT[6] and large German BERT[7] (*German* BERT). In preliminary experiments we also used publicly available German BERT models (deepset[8] and DBMDZ[9]). Since their performance was not convincing we did not include them in our final setup.

The *Europeana* BERT data comes from the Europeana Newspapers collection[10], which contains historical news articles in 12 languages published between 1618 and 1990. The *Europeana* BERT model was trained on 51GB of newspapers, extracted from German Europeana. It mainly covers newspaper articles from the 18th to 20th century. *German* BERT was trained on a huge collection of various historical resources.

3 Methods

To develop an efficient NER tagger for historical texts we experiment with stacking methods described in the following.

We experiment with different kinds of ensembling/stacking approaches on the development set to figure out the optimal combination of embeddings and language models. Our final system CISTERIA uses an ensemble of word embeddings, transformer-based language models and FLAIR embeddings. To arrive at the best combination of embeddings for CISTERIA we conduct experiments where we a) select the best word embeddings, FLAIR embeddings and transformer-based language models independently and b) combine the best selected word embedding, the best transformer-based language model and the best FLAIR embeddings and feed those to our network. The network for the classification is a bidirectional LSTM with a conditional random field (CRF) as final output layer as proposed by [14]. Note that we train separate models for the metonymic and the literal sense span.

4 Implementation and Experiments

The following describes the implementation of our approach, overviews the different experiments and presents the results. Our final system for the CLEF HIPE 2020 evaluation lab is referred to as CISTERIA.

To feed the CLEF-HIPE data into our tagger we need several preprocessing steps. Our preprocessing includes sentence splitting (rule based method) and normalizing word hyphenations. The motivation behind normalizing hyphenation is that pretrained language models normally include normalized text and the word hyphenation character in the CLEF-HIPE shared task is a special symbol (¬)

[6] https://github.com/stefan-it/europeana-bert
[7] Under review.
[8] https://huggingface.co/bert-base-german-cased
[9] https://github.com/dbmdz/berts
[10] http://www.europeana-newspapers.eu/

and does not occur in training corpora for pretrained language models. As we use contextualized word embeddings, the correct hyphenation is very important to produce high quality embeddings. To get the data ready for evaluation with the officially provided evaluation script, we perform a reverse process and add word hyphenation and sentence boundaries again.

We use the Flair [1] library to train our NER tagging models and we make use of Bert embeddings in a *feature-based* setting. In order to get a representation for an input token, we first compute the mean of the first subword over all layers of the transformer-based architecture and feed the resulting representation into a bidirectional LSTM with a CRF as the final layer, following [3]. To ensemble different embeddings and language models their representations are concatenated and the resulting vector is processed by the neural model. Cisteria was trained on the official training and development data and does not use any other additional labeled training data.

For the experiments with transformer-based language models, we fine-tune Bert models using the Hugging Face Transformers library [29]. For these fine-tuning experiments, we use a batch size of 16 and train 10 epochs. We perform three runs per transformer-based model and select the best model based on development F1-score. We do not perform extensive hyperparameter search.

We then use the fine-tuned model in Flair (feature-based approach) for all further experiments. We use a bidirectional LSTM with 256 hidden states and a batch size of 16. The original Bert paper [8] uses the last four layers of the transformer-based model for a feature-based NER model. Additionally, we reduce the learning rate by a factor of 0.5 with a patience of 3. This factor determines the number of epochs with no improvement after which the learning rate will be reduced and can be seen as early stopping.

We found that fine-tuning a Bert model for the metonymic sense span was very unstable resulting in zero F1-scores. This is a well known problem for datasets when only a small number of training instances are available and a solution could be to use a different dropout strategy [17]. For that reason we trained a model using the CLEF-HIPE Flair embeddings. In the prediction phase we only do predictions when an entity is detected for the literal sense span.

Our final system for the literal sense span uses FastText embeddings trained on Wikipedia (*FastText Wiki*) and a self-trained large *German* Bert model. For the metonymic sense span we train a separate model that uses FastText embeddings trained on Wikipedia and Flair embeddings provided by the organizers.

4.1 Results

For the evaluation of NER there are two regimes: strict and fuzzy. The strict regime corresponds to exact boundary matching whereas the fuzzy takes overlapping boundaries into account, a detailed description can be found in [11]. In

addition spans are evaluated w.r.t literal or metonymic sense (see section 1.1). We evaluate our systems using the official evaluation script[11].

All our reported results on the development set refer to the F1 score for coarse grained NER in the strict scenario for the literal sense. For the test set we report precision, recall and F1 score for both scenarios in the literal sense as well as in the metonymic sense (see Table 8). According to the overview paper of the shared task [11] the baseline in the strict evaluation scenario for German Coarse NER in literal sense results in **47.6%** F1-score (see Table 7).

Our results of the experiments with different word embeddings show that the *FastText Wiki* embeddings perform best, see Table 3. With an F1-score of approx. 69% they can overcome the baseline by more than 20 percentage points. Interesting is that the *FastText Wiki* embeddings are not trained on the biggest amount of data compared to the other word embeddings (see Table 2).

Model	F1
FastText Wiki	**69.28** ±0.65
FastText CC	66.38 ±0.51
BPEmb [12]	67.71 ±0.48
MultiBPEmb [13]	66.22 ±0.14

Table 3. Experiments with different word Embeddings on German development set. Averaged F1-score over 3 runs is reported here. Best result in bold.

Different Flair embeddings lead consistently to better results than using word embeddings. The Flair embeddings provided by the organizers (*CLEF-HIPE*) perform best, with an F1-score of 77.04% (see Table 4). The gap between the different Flair embeddings is comparably large and ranges from seven to three percentage points difference. Here the embeddings that were trained on the biggest amount of data perform best and the *Redewiedergabe* embeddings that were trained on the least amount perform worst.

Model	F1
Hamburger Anzeiger [23]	74.14 ±0.11
Wiener Zeitung [23]	75.07 ±0.11
Redewiedergabe [6]	70.21 ±0.27
German (Flair) [3]	74.98 ±0.30
CLEF-HIPE	**77.04** ±0.12

Table 4. Experiments with different Flair Embeddings on German development set. Averaged F1-score over 3 runs is reported here. Best result in bold.

[11] `https://github.com/impresso/CLEF-HIPE-2020-scorer`

Model	F1
Europeana Bert (cased)	80.41 ±0.14
Europeana Bert (uncased)	79.66 ±0.32
German Bert (cased, large)	**82.11** ±0.50

Table 5. Experiments with different Bert models on German development set. Averaged F1-score over 3 runs is reported here. Best result in bold.

The usage of Bert enhances the performance once more. The *German* Bert model performs best and results in 82.11% F-score (see Table 5). Again this is the model that was trained on the biggest amount of data. The cased version of *Europeana* Bert leads to a similar performance with approx. two percentage points less. Since German is case sensitive it is understandable that the cased models perform better than the uncased ones. Like with the Flair embeddings every setup with Bert outperforms the models of our previous experiments.

Model	F1
FastText (Wikipedia) + CLEF-HIPE + *German* Bert	83.57 ±0.36
FastText (Wikipedia) + CLEF-HIPE	77.97 ±0.47
FastText (Wikipedia) + *German* Bert	**83.69** ±0.08

Table 6. Experiments with different stacking experiments on German development set. Averaged F1-score over 3 runs is reported here. Best result in bold.

Finally the combination of *German* Bert with the *FastText Wiki* embeddings outperforms all of our other systems on the development set and results in 83.69% (see Table 6). This result is plausible if we compare it to the best F1-scores of [16] on other historical datasets. For two datasets their performance is around 84%. The addition of the best Flair embeddings decreases the results slightly. If combining the best Flair embeddings with the best FastText embeddings the model performs better than using Flair embeddings only but still worse than the other stacking approaches. The performance of our best system is approx. 40% better than the baseline, which is a large improvement.

4.2 Discussion of Results

We want to relate our final results on the test set to those of the other participating teams. Compared to the baseline our final systems (CISTERIA) could perform very good. If we take a look at the median of all participating teams our system for the literal sense performs approx. 2% points better in the strict scenario and is almost on par with the median in the fuzzy scenario (see Table 7). For both regimes the best system *L3i* [5] outperforms ours by slightly more than

10% points. This could be due to the fact that they use powerful transformer-based embeddings for different languages and a hierarchical transformer-based attention model [28] together with a multi task learning setting approach. Our experiments with BERT embeddings show that the model can benefit from the German Europeana BERT language model a lot and that only a model trained with even more data could outperform it. Therefore it is not surprising that a model trained with more of these powerful BERT embeddings performs even better. The benefit of the combination of models for different languages is at hand and we suppose that our model performances can be enhanced if we integrate multilinguality as well.

Team	Strict			Fuzzy		
	P	R	F1	P	R	F1
CISTERIA	<u>0.745</u>	0.578	0.651	**0.880**	0.683	0.769
EHRMAMA [27]	0.697	<u>0.659</u>	<u>0.678</u>	0.814	0.765	0.789
L3I [5]	**0.790**	**0.805**	**0.797**	<u>0.870</u>	**0.886**	**0.878**
SBB [15]	0.499	0.484	0.491	0.730	0.708	0.719
SINNER [21]	0.658	0.658	0.658	0.775	<u>0.819</u>	<u>0.796</u>
UPB [7]	0.677	0.575	0.621	0.788	0.740	0.763
UVA-ILPS [22]	0.499	0.556	0.526	0.689	0.768	0.726
WEBIS [26]	0.695	0.337	0.454	0.833	0.405	0.545
Baseline	0.643	0.378	0.476	0.790	0.464	0.558
Median	0.686	0.576	0.636	0.801	0.752	0.766

Table 7. Results for NERC-Coarse literal with micro precision, recall and F1-score on the test set. Bold font indicates highest, underlined the second highest result.

In the evaluation w.r.t the metonymic sense it turns out that our approach to train a separate model was constructive. In both regimes our system performs clearly above the median and in the fuzzy regime our F1-score is the second best (see Table 8). Again the *L3i* system can reach the best scores, probably due to the same reasons as mentioned above. Our results support our strategy that we only do predictions for tokens where the literal sense is classified as an entity.

Regarding the precision our system performs very well and reaches second best performance in all cases, except for the fuzzy evaluation in the literal sense where our system performs best. Unfortunately the recall is relatively low with around 50% for the metonymic sense and 57%/68% for the strict/fuzzy evaluation in the literal sense. Our system has the ability to classify correctly if it identifies a token as a possible entity but has problems with finding the entities as such.

Team	Strict			Fuzzy		
	P	R	F1	P	R	F1
Cisteria	0.738	0.500	0.596	0.787	0.534	0.636
Ehrmama [27]	0.696	0.542	0.610	0.707	0.551	0.619
L3i [5]	0.571	**0.712**	**0.634**	0.626	**0.780**	**0.694**
Baseline	**0.814**	0.297	0.435	**0.814**	0.297	0.435

Table 8. Results for NERC-Coarse metonymic with micro precision, recall and F1-score. Bold font indicates highest, underlined the second highest result.

5 Future Work

The approach of the winning team suggests to include multilingual language models and/or more data. Since a lot of powerful pretrained language models are available we will integrate some of them in Cisteria.

Another strategy is to take into account the domain of historical language even more. Since there is a lot of noise in the data due to OCR it greatly differs from modern standard language. Nevertheless there are many modern corpora available on which transformer-based language models can be trained. Our goal is to increase the similarity of those modern corpora to historical data. Therefore we want to recreate some of the phenomena in historical corpora in the modern corpora that we use for training the language models.

Besides that, manual rule-based sentence segmentation could have drawbacks (e.g. bad segmentation could lead to short sentences). So in future experiments we could use the context before and after the actual training sentence, such as in [18]. This approach could eliminate potential drawbacks of an automatically sentence segmented training corpus, because shorter sentences are now enhanced with longer contexts.

6 Conclusion

We proposed a system to solve coarse grained NER for German in the CLEF HIPE shared task. We conducted experiments with ensembling different word and subword embeddings as well as transformer-based language models on the basis of a bidirectional LSTM with a CRF as final layer. To use historical resources at best we trained large language models on historical German data, such as the German Europeana collection. Our best system uses FastText embeddings trained on German Wikipedia data in combination with a large German Bert language model. With a performance of 65.1% F1-score our best system performs slightly better than the median in the strict scenario for the literal sense and with an F1-score of 76.9% on par with the median in the fuzzy scenario. For the metonymic sense our best system performs clearly above the baseline and reaches the second best performance in the fuzzy scenario.

References

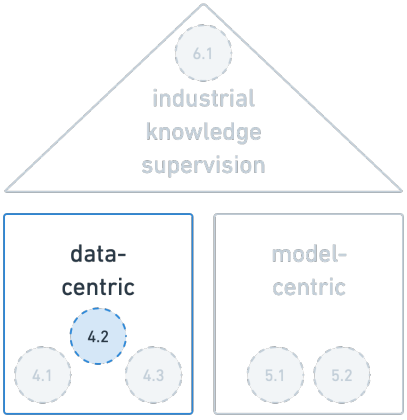
1. Akbik, A., Bergmann, T., Blythe, D., Rasul, K., Schweter, S., Vollgraf, R.: FLAIR: An Easy-to-Use Framework for State-of-the-Art NLP. In: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations). pp. 54–59. Association for Computational Linguistics, Minneapolis, Minnesota (Jun 2019), https://www.aclweb.org/anthology/N19-4010
2. Akbik, A., Bergmann, T., Vollgraf, R.: Pooled Contextualized Embeddings for Named Entity Recognition. In: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers). pp. 724–728. Association for Computational Linguistics, Minneapolis, Minnesota (Jun 2019), https://www.aclweb.org/anthology/N19-1078
3. Akbik, A., Blythe, D., Vollgraf, R.: Contextual String Embeddings for Sequence Labeling. In: Proceedings of the 27th International Conference on Computational Linguistics. pp. 1638–1649. Association for Computational Linguistics, Santa Fe, New Mexico, USA (Aug 2018), https://www.aclweb.org/anthology/C18-1139
4. Baevski, A., Edunov, S., Liu, Y., Zettlemoyer, L., Auli, M.: Cloze-driven Pretraining of Self-attention Networks. In: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP). Association for Computational Linguistics, Hong Kong, China (Nov 2019), https://www.aclweb.org/anthology/D19-1539
5. Boros, E., Linhares Pontes, E., Cabrera-Diego, L.A., Hamdi, A., Moreno, J.G., Sidère, N., Doucet, A.: Robust Named Entity Recognition and Linking on Historical Multilingual Documents. In: Cappellato, L., Eickhoff, C., Ferro, N., Névéol, A. (eds.) CLEF 2020 Working Notes. Working Notes of CLEF 2020 - Conference and Labs of the Evaluation Forum. CEUR-WS (2020)
6. Brunner, A., Engelberg, S., Jannidis, F., Tu, N.D.T., Weimer, L.: Corpus REDEWIEDERGABE. In: Proceedings of The 12th Language Resources and Evaluation Conference. pp. 803–812. European Language Resources Association, Marseille, France (May 2020), https://www.aclweb.org/anthology/2020.lrec-1.100
7. Craita, C.C., Cercel, D.C.: Multilingual Named Entity Recognition on Historical Texts Using Transfer and Multi-Task Learning. In: Cappellato, L., Eickhoff, C., Ferro, N., Névé ol, A. (eds.) CLEF 2020 Working Notes. Working Notes of CLEF 2020 - Conference and Labs of the Evaluation Forum. CEUR-WS (2020)
8. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers). pp. 4171–4186. Association for Computational Linguistics, Minneapolis, Minnesota (Jun 2019), https://www.aclweb.org/anthology/N19-1423
9. Ehrmann, M., Romanello, M., Flückiger, A., Clematide, S.: HIPE - Shared Task Participation Guidelines (v1.1) (2020). https://doi.org/10.5281/zenodo.3677171
10. Ehrmann, M., Romanello, M., Flückiger, A., Clematide, S.: Impresso Named Entity Annotation Guidelines (Jan 2020). https://doi.org/10.5281/zenodo.3604227
11. Ehrmann, M., Romanello, M., Flückiger, A., Clematide, S.: Overview of CLEF HIPE 2020: Named Entity Recognition and Linking on Historical Newspapers. In:

Arampatzis, A., Kanoulas, E., Tsikrika, T., Vrochidis, S., Joho, H., Lioma, C., Eickhoff, C., Névéol, A., Cappellato, L., Ferro, N. (eds.) Experimental IR Meets Multilinguality, Multimodality, and Interaction. Proceedings of the 11th International Conference of the CLEF Association (CLEF 2020). Lecture Notes in Computer Science (LNCS), vol. 12260. Springer (2020)

12. Heinzerling, B., Strube, M.: BPEmb: Tokenization-free Pre-trained Subword Embeddings in 275 Languages. In: chair), N.C.C., Choukri, K., Cieri, C., Declerck, T., Goggi, S., Hasida, K., Isahara, H., Maegaard, B., Mariani, J., Mazo, H., Moreno, A., Odijk, J., Piperidis, S., Tokunaga, T. (eds.) Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018). European Language Resources Association (ELRA), Miyazaki, Japan (May 7-12, 2018 2018)
13. Heinzerling, B., Strube, M.: Sequence Tagging with Contextual and Non-Contextual Subword Representations: A Multilingual Evaluation. In: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics. pp. 273–291. Association for Computational Linguistics, Florence, Italy (Jul 2019), https://www.aclweb.org/anthology/P19-1027
14. Huang, Z., Xu, W., Yu, K.: Bidirectional LSTM-CRF models for sequence tagging. arXiv preprint arXiv:1508.01991 (2015)
15. Labusch, K., Neudecker, C.: Named Entity Disambiguation and Linking Historic Newspaper OCR with BERT. In: Cappellato, L., Eickhoff, C., Ferro, N., Névéol, A. (eds.) CLEF 2020 Working Notes. Working Notes of CLEF 2020 - Conference and Labs of the Evaluation Forum. CEUR-WS (2020)
16. Labusch, K., Neudecker, C., Zellhöfer, D.: Bert for named entity recognition in contemporary and historic german. In: Proceedings of the 15th Conference on Natural Language Processing (KONVENS 2019): Long Papers. pp. 1–9. German Society for Computational Linguistics & Language Technology, Erlangen, Germany (2019)
17. Lee, C., Cho, K., Kang, W.: Mixout: Effective Regularization to Finetune Large-scale Pretrained Language Models. In: International Conference on Learning Representations (2020), https://openreview.net/forum?id=HkgaETNtDB
18. Luoma, J., Pyysalo, S.: Exploring Cross-sentence Contexts for Named Entity Recognition with BERT. arXiv e-prints arXiv:2006.01563 (Jun 2020)
19. Mikolov, T., Grave, E., Bojanowski, P., Puhrsch, C., Joulin, A.: Advances in Pre-Training Distributed Word Representations. In: Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018) (2018)
20. Neudecker, C.: An Open Corpus for Named Entity Recognition in Historic Newspapers. In: Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16). pp. 4348–4352. European Language Resources Association (ELRA), Portorož, Slovenia (May 2016), https://www.aclweb.org/anthology/L16-1689
21. Ortiz, S., Pedro, J., Dupont, Y., Lejeune, G., Tian, T.: SinNer@Clef-Hipe2020: Sinful adaptation of SotA models for Named Entity Recognition in historical French and German newspapers. In: Cappellato, L., Eickhoff, C., Ferro, N., Névé ol, A. (eds.) CLEF 2020 Working Notes. Working Notes of CLEF 2020 - Conference and Labs of the Evaluation Forum. CEUR-WS (2020)
22. Provatorova, V., Vakulenko, S., Kanoulas, E., Dercksen, K., van Hulst, J.M.: CLEF HIPE Working Notes: UvA ILPS & REL. In: Cappellato, L., Eickhoff, C., Ferro, N., Névé ol, A. (eds.) CLEF 2020 Working Notes. Working Notes of CLEF 2020 - Conference and Labs of the Evaluation Forum. CEUR-WS (2020)

23. Schweter, S., Baiter, J.: Towards Robust Named Entity Recognition for Historic German. In: Proceedings of the 4th Workshop on Representation Learning for NLP (RepL4NLP-2019). pp. 96–103. Association for Computational Linguistics, Florence, Italy (Aug 2019), https://www.aclweb.org/anthology/W19-4312
24. Sennrich, R., Haddow, B., Birch, A.: Neural Machine Translation of Rare Words with Subword Units. In: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). pp. 1715–1725. Association for Computational Linguistics, Berlin, Germany (Aug 2016), https://www.aclweb.org/anthology/P16-1162
25. Souza, F., Nogueira, R., Lotufo, R.: Portuguese Named Entity Recognition using BERT-CRF (2019)
26. Tobollik, T., Wiegmann, M., Wolska, M., Stein, B.: Enrichement-based Oversampling for Coarse-grained NER in Historical Text. In: Cappellato, L., Eickhoff, C., Ferro, N., Névé ol, A. (eds.) CLEF 2020 Working Notes. Working Notes of CLEF 2020 - Conference and Labs of the Evaluation Forum. CEUR-WS (2020)
27. Todorov, K., Colavizza, G.: Transfer Learning for Named Entity Recognition in Historical Corpora. In: Cappellato, L., Eickhoff, C., Ferro, N., Névéol, A. (eds.) CLEF 2020 Working Notes. Working Notes of CLEF 2020 - Conference and Labs of the Evaluation Forum. CEUR-WS (2020)
28. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention Is All You Need. CoRR **abs/1706.03762** (2017), http://arxiv.org/abs/1706.03762
29. Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Le Scao, T., Gugger, S., Drame, M., Lhoest, Q., Rush, A.M.: HuggingFace's Transformers: State-of-the-art Natural Language Processing. arXiv e-prints arXiv:1910.03771 (Oct 2019)

4.2. Data Centric Domain Adaptation for Historical Text with OCR Errors



Contributing Article

März, L., Schweter, S., Poerner, N., Roth, B., & Schütze, H. (2021). Data Centric Domain Adaptation for Historical Text with OCR Errors. In *International Conference on Document Analysis and Recognition* (pp. 748-761). Springer, Cham.
`https://doi.org/10.1007/978-3-030-86331-9_48`

Copyright Information

Author Contributions

The idea and conception for the work were brought up in joint discussions between all authors. Programming was mainly done by Luisa März and included data compilation, data pre-processing, data corruption, data modification based on the corruption and perturbation algorithms, model training, integration of pre-trained embeddings, as well as prediction, evaluation, and analysis. Stefan Schweter contributed the pre-trained historical language models and advised on the implementation. Nina Poerner contributed large parts of the perturbation algorithm. Luisa März and Stefan Schweter put in substantial effort to create a transparent and fully reproducible preparation of the released data sets. The paper was mainly written by Luisa März and constantly revised by all authors. All authors finally revised and proofread the manuscript.

Supplementary Material

`https://github.com/stefan-it/historic-domain-adaptation-icdar`

Data Centric Domain Adaptation for Historical Text with OCR Errors

Luisa März[1,2,4( )], Stefan Schweter[3], Nina Poerner[1], Benjamin Roth[2], and Hinrich Schütze[1]

[1] Center for Information and Language Processing, Ludwig Maximilian University, Munich, Germany
maerz@cis.lmu.de, inquiries@cislmu.org
[2] Digital Philology, Research Group Data Mining and Machine Learning, University of Vienna, Vienna, Austria
[3] Bayerische Staatsbibliothek München, Digital Library/Munich Digitization Center, Munich, Germany
[4] NLP Expert Center, Data:Lab, Volkswagen AG, Munich, Germany
https://www.cis.uni-muenchen.de/

Abstract. We propose new methods for in-domain and cross-domain Named Entity Recognition (NER) on historical data for Dutch and French. For the cross-domain case, we address domain shift by integrating unsupervised in-domain data via contextualized string embeddings; and OCR errors by injecting synthetic OCR errors into the source domain and address data centric domain adaptation. We propose a general approach to imitate OCR errors in arbitrary input data. Our cross-domain as well as our in-domain results outperform several strong baselines and establish state-of-the-art results. We publish preprocessed versions of the French and Dutch Europeana NER corpora.

Keywords: Named Entity Recognition · Historical data · Flair

1 Introduction

Neural networks achieve good NER accuracy on high-resource domains such as modern news text or Twitter [2,4]. But on historical text, NER often performs poorly. This is due to several challenges: i) Domain shift: Entities in historical texts can be different from contemporary entities, this makes it difficult for modern taggers to work with historical data. ii) OCR errors: historical texts – usually digitized by OCR – contain systematic errors not found in non-OCR text [14]. In addition these errors can change the surface form of entities. iii) Lack of annotation: Some historical text is now available in digitized form, but without labels, and methods are required for beneficial use of such data [16].

In this paper, we address *data centric domain adaptation* for NER tagging on historical French and Dutch data. Following Ramponi and Plank [20], data centric approaches do not adapt the model but the training data in order to

J. Lladós et al. (Eds.): ICDAR 2021, LNCS 12822, pp. 748–761, 2021.
https://doi.org/10.1007/978-3-030-86331-9_48

improve generalization across domains. We address both in-domain and cross-domain NER. In the cross-domain setup, we use supervised contemporary data and integrate unsupervised historical data via contextualized embeddings. We introduce artificial OCR errors into supervised modern data and find a way to perturb corpora in a general and robust way – independent of language or linguistic properties.

In the cross-domain setup as well as in-domain, our system outperforms neural and statistical state-of-the-art methods, achieving 69.3% F_1 for French and 63.4% for Dutch. With the in-domain setup, we achieve 77.9% for French and 84.2% for Dutch. If we only consider named entities that contain OCR errors, our domain-adapted cross-domain tagger even performs better (83.5% French/ 46.2% Dutch) than in-domain training (77.1% French/ 43.8% Dutch). Our main contributions are:

- Release of the preprocessed French and Dutch NER corpora[1];
- Developing synOCR to mimic historical data while exploiting the annotation of modern data;
- Training historical embeddings on a large amount of unlabeled historical data;
- Ensembling a NER system that establishes SOTA results for both languages and scenarios.

2 Methods

2.1 Architecture

We use the Flair NLP framework [1]. Flair taggers achieve SOTA results on various benchmarks and are well suited for NER. Secondly, there are powerful Flair embeddings. They are trained without explicit notion of words and model words as character sequences depending on their context. These two properties contribute to making atypical entities - even those with distorted surface - easier to recognize. In all our experiments, word embeddings are generated by a character-level RNN and passed to a word-level bidirectional LSTM with a CRF as the final layer. Depending on the experiment, we concatenate ($\odot$) additional embeddings and refer to that as *ensembling process*.

2.2 Noise Methods

Since digitizing by OCR introduces a lot of noise into the data, we recreate some of those phenomena in the modern corpora that we use for training. Our goal is to increase the similarity of historical (OCR'd) and modern (clean) data. An example drawn from the dutch training corpora can be found in Fig. 1. Words that are different from the original text are indicated in bold font.

Generation of Synthetic OCR (synOCR) Errors. This method processes every sentence by assigning a randomly selected font and a font size between 6

[1] https://github.com/stefan-it/historic-domain-adaptation-icdar

and 11 pt. Batches of 150 sentences are printed to PDF documents and then converted to PNG images. The images are perturbed using imgaug[2] with the following steps: (i) rotation, (ii) Gaussian blur and (iii) white or black pixel dropout. The resulting image is recognized using tesseract version 0.2.6.[3]. We re-align the recognized sentences with the clean annotated corpus to transfer the NER tags. For the alignment between original and degraded text we select a window of the bitext and calculate a character-based alignment cost. We then use the Wagner-Fisher algorithm [27] to obtain the best alignment path through the window and the lowest possible cost. If the cost is below a threshold, we shift the window to the mid-point of the discovered path. Otherwise, we iteratively increase the window size and re-align, until the threshold criterion is met. This procedure allows us to find an alignment with reasonable time and space resources, without risking to lose the optimal path in low-quality areas. Finally this results in an OCR-error enhanced annotated corpus with a range of recognition quality, from perfectly recognized to fully illegible. We refer to OCR-corrupted data as *synOCR'd data*.

Generation of Synthetic Corruptions. This method is applied to our modern corpora, again to introduce noise as we find it in historical data. Similar to [21], we randomly corrupt 20% of all words by (i) inserting a character or (ii) removing a character or (iii) transposing two characters. Therefore, we use the standard alphabet of French/Dutch. We re-align the corrupted tokens with the clean annotated tokens while maintaining the sentence boundaries to transfer the NER tags. Since the corruption method does not break the word boundaries we can simply map each corrupted word to the original one and retrieve the corresponding NER tag. We refer to synthetically corrupted data as *corrupted data*.

De tekst van het arrest is nog niet schriftelijk beschikbaar maar het bericht werd alvast bekendgemaakt door een communicatiebureau dat Floralux inhuurde .	*Original*
De **:eks:** van het attest is nog **hie: schiiizeujk beschikbaaz** maar **bet helicht weéd alvas:** bekendqemaakt door **sen‘ communicacisbuxeau dac** Floralux inhuurde **, ' *———**	*synOCR*
De tekst van het arrest is nog niet schriftelijk beschikbaar maar **ht bCericht** werd alvast bekendgemaakt door **eeRn** communicatiebureau dat Floralux inhuurde .	*corrupted*

Fig. 1. Example from the dutch train set. Text in its original, the synOCR'd and the corrupted form.

[2] https://github.com/aleju/imgaug
[3] https://github.com/tesseract-ocr/

2.3 Embeddings

We experiment with various common embeddings and integrate them in our neural system. Some of them are available in the community and some others we did train on data described in Sect. 3.1.

Flair Embeddings. [3] present contextual string embeddings which can be extracted from a neural language model. Flair embeddings use the internal states of a trained character language model at token boundaries. They are contextualized because a word can have different embeddings depending on its context. These embeddings are also less sensitive to misspellings and rare words and can be learned on unlabeled corpora. We also use *multilingual* Flair *embeddings*. They were trained on a mix of corpora from different domains (Web, Wikipedia, Subtitles, News) and languages.

Historical Embeddings. We train Flair embeddings on large unlabeled historical corpora from a comparable time period (see Sect. 3.1) and refer to them as *historical embeddings*.

BERT Embeddings. Since *BERT embeddings* [9] produce state-of-the-art results for a wide range of NLP tasks, we also experiment with multilingual BERT embeddings[4]. BERT embeddings are subword embeddings based on a bidirectional transformer architecture and can model the context of a word. For NER on CoNLL-03 [25], BERT embeddings do not perform as well as on other tasks [9] and we want to examine if this observation holds for a cross-domain scenario with different data.

FastText Embeddings. We do also use *FastText embeddings* [6] which are widely used in NLP. They can be efficiently trained and address character-level phenomena. Subwords are used to represent the target word (as a sum of all its subword embeddings). We use pre-trained FastText embeddings for French/Dutch[5].

Character-Level Embeddings. Due to the OCR errors out-of-vocabulary problems occur. Lample et al. [15] create *character embeddings*, passing all characters in a sentence to a bidirectional LSTM. To obtain word representations, the forward and backward representations of all the characters of the word from this LSTM are concatenated. Having the character embedding, every single words vector can be formed even if it is out-of-vocabulary. Therefore, we do also compute these embeddings for our experiments.

3 Experiments

In the cross-domain setup, we train on modern data (clean or synOCR'd) and test on historical data (OCR'd). In the in-domain setup, we train and test on a set of historical data (OCR'd). We do use different combinations of embeddings and also use our noise methods in the experiments.

[4] We use the cased variant from https://huggingface.co/bert-base-multilingual-cased
[5] https://fasttext.cc/docs/en/crawl-vectors.html

3.1 Data

We use different data sources for our experiments from which some are openly available and some historical data come from an in-house project. For an overview of different properties (domain, labeling, size, language) see Table 1.

Annotated Historical Data. Our annotated historical data comes from the Europeana Newspapers collection[6], which contains historical news articles in 12 languages published between 1618 and 1990. Parts of the German, Dutch and French data were manually annotated with NER tags in IO/IOB format for PER (person), LOC (location), ORG (organization) by Neudecker [17]. Each NER corpus contains 100 scanned pages (with OCR accuracy over 80%), amounting to 207K tokens for French and 182K tokens for Dutch.

We preprocess the data as follows. We perform sentence splitting, filter out metadata, re-tokenize punctuation and convert all annotations to IOB1 format. We split the data 80/10/10 into train/dev/test. We will make this preprocessed version available in CoNLL format.

Annotated Modern Data. For the French cross-domain experiments, we use the French WikiNER corpus [18]. WikiNER is tagged in IOB format with an additional MISC (miscellaneous) category; we convert the tags to our Europeana format. For better comparability we downsample (sentence-wise) the corpus from 3.5M to 525K tokens. Therefore, entire sentences were sampled uniformly at random without replacement. For Dutch, we use the CoNLL-02 corpus [24], which consists of four editions of the Belgian Dutch newspaper "De Morgen" from the year 2000. The data comprises 309K tokens and is annotated for PER, ORG, LOC and MISC. We convert the tags to our Europeana format.

Unlabeled Historical Data. For historical French, we use "Le Temps", a journal published between 1861 and 1942 (initially under a different name), a similar time period as the Europeana Newspapers. The corpus contains 977M tokens and is available from the National Library of France.[7] For historical Dutch, we use data from an in-house OCR project. The data is from the 19th century and it consists of 444M tokens. We use the unlabeled historical data to pre-train historical embeddings (see Sect. 2.3).

3.2 Baselines

We experiment with three baselines. (i) The Java implementation[8] of the Stanford NER tagger [12]. (ii) A version of Stanford NER published by Neudecker [17][9] that was trained on Europeana. In contrast to our system they trained

[6] http://www.europeana-newspapers.eu/
[7] https://www.bnf.fr/fr
[8] https://nlp.stanford.edu/software/CRF-NER.html
[9] https://lab.kb.nl/dataset/europeana-newspapers-ner

Table 1. Number of tokens per dataset in our experiments.

	Domain	Data	Labeled	Size
French	Historical	Europeana NER	+	207K
	Modern	WikiNER	+	525K
	Historical	"Le Temps"	−	977M
Dutch	Historical	Europeana NER	+	182K
	Modern	CoNLL-02	+	309K
	historical	in-house OCR	−	444M

theirs on the entire amount of the labeled Europeana corpora with 4-fold cross validation. (iii) NN base. The neural network (see Sect. 2.1) with FastText, character and multilingual Flair embeddings, as recommended in Akbik et al. [1]. For French, we also list the result reported by Çavdar [8]. Since we do not have access to their implementation and could not confirm that their data splits conform to ours, we could not compute the combined F_1 score or test for significance.

Table 2. Results (F_1 scores on French/Dutch Europeana test set) of training on Europeana French/Dutch training set. *Hist. Embs.* are historical embeddings. Scores marked with * are significantly lower than *NN base* ⊙ *hist. Es.*

French models	Overall	PER	ORG	LOC
Çavdar		0.68	0.37	0.68
Stanford NER tagger	0.662*	0.569*	0.335*	0.753*
Stanford Neudecker	0.750*	0.750*	**0.505**	0.826*
NN base	0.741*	0.703*	0.320*	0.813*
NN base ⊙ hist. Embs	**0.779**	**0.759**	0.498	**0.832**
Dutch models	Overall	PER	ORG	LOC
Stanford NER tagger	0.696*	0.640*	0.333*	0.794*
Stanford Neudecker	0.623*	0.700*	0.253*	0.702*
NN base	0.818*	0.809*	0.442*	0.871*
NN base ⊙ hist. Embs	**0.842**	**0.833**	**0.480**	**0.891**

4 Results and Discussion

We evaluate our systems using the CoNLL-2000 evaluation script[10], with F_1 score. To check statistical significance we use randomized testing [28] and results are considered significant if $p < 0.05$.

[10] https://www.clips.uantwerpen.be/conll2000/chunking/conlleval.txt

4.1 In-domain Setup

For both languages we achieve the best results with NN base ⊙ historical embeddings. With this setup we can produce F_1 scores of around 80% for both languages, which outperforms all three baselines in the overall performance significantly. The results are presented in Table 2. For French, the overall F_1 score as well as the F_1 for LOC and ORG is best with NN base ⊙ historical embeddings. For ORG the pre-trained tagger of Neudecker [17] works best, which could be due to the gazetteer information they included and of course due to the fact that they train with the entire Europeana data. We hypothesize that the category with the most structural changes over time is ORG. In the military or ecclesiastical context in particular, there are a number of names that no longer exist (in this form). For Dutch we observe the best overall performance with NN base ⊙ historical embeddings except for all entity types.

4.2 Cross-Domain Setup

As shown in Table 3, NN base performs better than the statistical Stanford NER baseline, which is in line with the observations for the in-domain training. We experimented with concatenating BERT embeddings to NN base. For both languages this increases the performance (Table 3, NN base ⊙ BERT). The usage of the historical embeddings is also very beneficial for both languages. We can achieve our best results by using BERT for Dutch and by using historical embeddings for French. We conclude that the usage of modern pre-trained language models is crucial for the performance of NER taggers.

We generated synthetic corruptions for the WikiNER/CoNLL corpus. This could not outperform NN base for both languages. The training on synOCR'd WikiNER/CoNLL gives slightly worse results than NN base too. The corruption of the training data without the usage of any embeddings seems to harm performance drastically, what is in line with the observation of Hamdi et al. [13]. It is striking that the training on corrupted/synOCR'd Dutch gives especially bad results for PER compared to French. A look at the Dutch test set shows that many entities are abbreviated first names (e.g. in ***A J*** *van Roozendal*) and are often misrecognized what leads to a performance decrease. For French the combination of NN base and historical embeddings, trained on corrupted data or on synOCR'd (*NN ensemble corrupted/ NN ensemble synOCR'd*) gives the best results and outperforms all other systems. For Dutch *NN ensemble corrupted* and *NN ensemble synOCR* give slightly worse results than NN base ⊙ BERT and NN base ⊙ historical embeddings, but performs better than the tagger trained on synOCR'd or corrupted data only (Table 3, NN ensemble).

Ablation Study. We analyze our results and examine the composition of NN ensemble synOCR more closely (since the results for NN ensemble corrupted are very similar we perform the analysis for NN ensemble synOCR as a representative for both NN ensemble).

The ablation study (see Table 4) shows that NN ensemble benefits from different information in combination. For French NN ensemble gives the best results only for PER. The overall performance increases if we do not use character level embeddings. There is a big performance loss if we omit the historical embeddings. If we do not train on synOCR'd data the performance decreases. For Dutch we can observe these facts even more clearly. If we do not train on synOCR'd data the F_1 score even increases. If omitting the historical embeddings we loose performance as well.

Table 3. Results of training on WikiNER/CoNLL corpus. Scores marked with * are significantly lower than *NN ensemble*.

French models	Overall	PER	ORG	LOC
Çavdar		0.48	0.11	0.56
Stanford NER tagger	0.536*	0.451*	0.059*	0.618*
NN base	0.646*	0.636*	0.096*	0.721*
NN base ⊙ BERT	0.660	0.639*	**0.163**	0.725*
NN base ⊙ hist. Embs.	0.672	0.661*	0.015*	0.748
Corrupted WikiNER	0.627*	0.635*	0.085*	0.710*
synOCR'd WikiNER	0.619*	0.590*	0.078	0.710
NN ensemble corrupted	**0.693**	0.624	0.063	**0.783**
NN ensemble synOCR	0.684	**0.710**	0.111	0.744
Dutch models	Overall	PER	ORG	LOC
Stanford NER tagger	0.371*	0.217*	0.083*	0.564*
NN base	0.567*	0.493*	0.085*	0.700*
NN base ⊙ BERT	**0.634**	**0.572**	**0.250**	0.771
NN base ⊙ hist. Embs.	0.632	0.568	0.084	0.738*
Corrupted CoNLL	0.535*	0.376*	0.155*	0.717*
synOCR'd CoNLL	0.521*	0.327*	0.061*	0.721*
NN ensemble corrupted	0.606*	0.439*	0.158	**0.799**
NN ensemble synOCR	0.614	0.481	0.157	0.775

To find out why our implementation of the assumption – synOCR increases the similarity of the data and improves results – does not have the expected effect, we analyze the test sets. It shows, that only 10% of the French and 6% of the Dutch entities contain OCR errors. Therefore the wrong predictions are mostly not due to the OCR errors, but due to the inherent difficulty of recognizing entities cross-domain. This also explains why synthetic noisyfication does not consistently improve the system. In addition there are some illegible lines in the synOCR'd corpora consisting of dashes and metasymbols, what is not similar to real OCR errors.

Table 4. Ablation study. Results of training on the clean and the synOCR'd WikiNER/CoNLL corpus.

French models	Overall	PER	ORG	LOC
NN ensemble synOCR	0.684	**0.710**	0.011	0.744
- char	**0.693**	0.681	0.078	**0.758**
- word	0.686	0.664	0.080	0.756
- hist. Embs.	0.619	0.590	0.078	0.710
- synOCR'd data	0.672	0.661	**0.015**	0.748
Dutch models	Overall	PER	ORG	LOC
NN ensemble synOCR	0.614	0.382	**0.157**	0.775
- char	0.600	0.404	0.119	**0.780**
- word	0.584	0.430	0.102	0.745
- hist. Embs.	0.521	0.327	0.061	0.721
- synOCR'd data	**0.632**	**0.568**	0.084	0.738

D' **Arras** <LOC> d´où nous avons débouché , nous sommes arrivés aux premières maisons de **Siiint-lûrenl-Blangy** <LOC>. *NN_ensemble*

D' **Arras** <LOC> d´où nous avons débouché , nous sommes arrivés aux premières maisons de **Siiint-lûrenl-Blangy** <O>. *Stanford NER tagger*

Fig. 2. Example sentence from the French test set.

To verify our assumption we also compare the different systems only on the entities with OCR errors. Here NN ensemble outperforms both of the cross-domain baselines (Table 5, Stanford NER tagger, NN base cross-domain). Compared to the French results Dutch is a lot worse. A look at the entities shows that in the Dutch test set there are many hyphenated words where both word parts are labeled. However, if looking at the parts of the word individually, a clear assignment to an entity type cannot be made, which leads to difficulties with tagging. Though it is plausible that NN ensemble can capture specific phenomena in the historical data better, since the difference between the domains is reduced by the synthetic noisyfication and the historical embeddings. The example in Fig. 2 drawn from the test set shows, that NN ensemble can handle noisy entities well in contrast to e.g. the Stanford NER tagger. Thus in a scenario with many OCR errors the NN ensemble performs well.

5 Related Work

There is some research on using natural language processing for improving OCR for historical documents [5, 26] and also on NER for historical documents [11]. In the latter - a shared task for Named Entity Processing in historical documents - Ehrmann et al. find that OCR noise drastically harms systems performance. Like

Table 5. Results on entities with OCR errors in the French/Dutch test set. Scores marked with * are significantly lower than *NN ensemble*.

Models	French	Dutch
Stanford NER tagger	0.661*	0.207*
NN base in-domain	0.771	0.438
NN base cross-domain	0.783	0.200*
NN ensemble synOCR	**0.835**	**0.462**

us several participants (e.g. [7,23]) also use language models that were trained on historical data to boost the performance of NER taggers. Schweter and Baiter [22] explore NER for historical German data in a cross-domain setting. Like us, they train a language model on unannotated in-domain data and integrate it into a NER tagger. In addition to the above mentioned work, we employ "OCR noisyfication" (Sect. 2.2) and examine the influence of different pretrained embeddings systematically. Çavdar [8] addresses NER and relation extraction on the French Europeana Newspaper corpus. Ehrmann et al. [10] investigate the performance of NER systems on Swiss historical Newspapers and show that historical texts are a great challenge compared to contemporary texts. They find that the LOC class entities causes the most difficulties in the recognition of named entities. The recent work of Hamdi et al. [13] investigates the impact of OCR errors on NER. To do so, they also perturb modern corpora synthetically with different degrees of error rates. They experiment with Spanish, Dutch and English. Like us they perturb the Dutch CoNLL corpus and train NER taggers on that data. Unlike us they do also train on a subset of the perturbed corpus. We test on a subset of the Dutch Europeana corpus. Hamdi et al. [13] show that neural taggers perform better compared to other taggers like the Stanford NER tagger and they also prove that performance decreases drastically if the OCR error rate increases. Piktus et al. [19] learn misspelling-oblivious FastText embeddings from synthetic misspellings generated by an error model for part-of-speech tagging. We use a similar corruption method, but we also use synOCR and historical embeddings for NER.

6 Conclusion

We proposed new methods for in-domain and cross-domain Named Entity Recognition (NER) on historical data and addressed data centric domain adaptation. For the cross-domain case, we handle domain shift by integrating non-annotated historical data via contextualized string embeddings; and OCR errors by injecting synthetic OCR errors into the modern data. This allowed us to get good results when labeled historical data is not available and the historical data is noisy. For training on contemporary corpora and testing on historical corpora we achieve new state-of-the-art results of 69.3% on French and 63.4% on Dutch. For the in-domain case we obtain state-of-the-art results of 77.9% for French and

84.2% for Dutch. There is an increasing demand for advancing the digitization of the world's cultural heritage. High quality digitized historical data, with reliable meta information, will facilitate convenient access and search capabilities, and allow for extensive analysis, for example of historical linguistic or social phenomena. Since named entity recognition is one of the most fundamental labeling tasks, it would be desirable that advances in this area translate to other labeling tasks in processing of historical data as well.

Acknowledgement. This work was funded by the European Research Council (ERC #740516).

A Appendix

Detailed Information About Experiments and Data
The computing infrastructure we use for all our experiments is one GeForce GTX 1080Ti GPU with an average runtime of 12 h per experiment. For the French and Dutch baseline model *NN base* we count 15,895,683 parameters each. For the French *NN ensemble* model there are 88,264,777 parameters and 96,895,161 parameters for the Dutch *NN ensemble*.

The Europeana Newspaper Corpus is split 80/10/10 into train/dev/test (Table 6). The downsampled French WikiNER corpus is split 70/15/15 into train/dev/test and the Dutch CoNLL-02 corpus is already split in its original version. The downloadable version of the data can be found here: https://github.com/stefan-it/historic-domain-adaptation-icdar.

Table 6. Number of tokens for each datasplit.

Dataset	Train	dev	Test
French Europeana	167,723	18,841	20,346
Dutch Europeana	147,822	16,391	18,218
French WikiNER	411,687	88,410	88,509
Dutch ConNLL-02	202,930	68,994	37,761

References

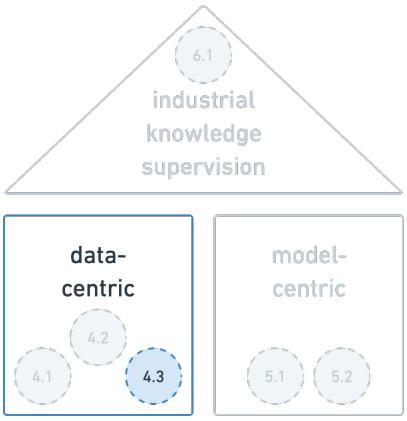
1. Akbik, A., Bergmann, T., Blythe, D., Rasul, K., Schweter, S., Vollgraf, R.: FLAIR: an easy-to-use framework for state-of-the-art NLP. In: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations), pp. 54–59. Association for Computational Linguistics, Minneapolis (June 2019). https://www.aclweb.org/anthology/N19-4010

2. Akbik, A., Bergmann, T., Vollgraf, R.: Pooled contextualized embeddings for named entity recognition. In: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Long and Short Papers), vol. 1, pp. 724–728. Association for Computational Linguistics, Minneapolis (June 2019)
3. Akbik, A., Blythe, D., Vollgraf, R.: Contextual string embeddings for sequence labeling. In: 27th International Conference on Computational Linguistics, COLING 2018, pp. 1638–1649 (2018)
4. Baevski, A., Edunov, S., Liu, Y., Zettlemoyer, L., Auli, M.: Cloze-driven pretraining of self-attention networks. In: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pp. 5359–5368. Association for Computational Linguistics, Hong Kong (November 2019). https://www.aclweb.org/anthology/D19-1539
5. Berg-Kirkpatrick, T., Durrett, G., Klein, D.: Unsupervised transcription of historical documents. In: Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 207–217. Association for Computational Linguistics, Sofia (August 2013). https://www.aclweb.org/anthology/P13-1021
6. Bojanowski, P., Grave, E., Joulin, A., Mikolov, T.: Enriching word vectors with subword information. Trans. Assoc. Comput. Linguist. **5**, 135–146 (2017). https://www.aclweb.org/anthology/Q17-1010
7. Boros, E., et al.: Robust named entity recognition and linking on historical multilingual documents. In: Conference and Labs of the Evaluation Forum (CLEF 2020). Working Notes of CLEF 2020 - Conference and Labs of the Evaluation Forum, vol. 2696, pp. 1–17. CEUR-WS Working Notes, Thessaloniki (September 2020). https://hal.archives-ouvertes.fr/hal-03026969
8. Çavdar, M.: Distant supervision for French relation extraction (2017)
9. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: pre-training of deep bidirectional transformers for language understanding. In: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Long and Short Papers), vol. 1, pp. 4171–4186. Association for Computational Linguistics, Minneapolis (June 2019). https://www.aclweb.org/anthology/N19-1423
10. Ehrmann, M., Colavizza, G., Rochat, Y., Kaplan, F.: Diachronic evaluation of NER systems on old newspapers. In: KONVENS (2016)
11. Ehrmann, M., Romanello, M., Flückiger, A., Clematide, S.: Extended overview of CLEF HIPE 2020: named entity processing on historical newspapers. Zenodo (October 2020)
12. Finkel, J.R., Grenager, T., Manning, C.: Incorporating non-local information into information extraction systems by Gibbs sampling. In: Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL 2005), pp. 363–370. Association for Computational Linguistics, Ann Arbor (June 2005). https://www.aclweb.org/anthology/P05-1045
13. Hamdi, A., Jean-Caurant, A., Sidère, N., Coustaty, M., Doucet, A.: Assessing and minimizing the impact of OCR quality on named entity recognition. In: Hall, M., Merčun, T., Risse, T., Duchateau, F. (eds.) TPDL 2020. LNCS, vol. 12246, pp. 87–101. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-54956-5_7
14. Jean-Caurant, A., Tamani, N., Courboulay, V., Burie, J.: Lexicographical-based order for post-OCR correction of named entities. In: 14th IAPR International

Conference on Document Analysis and Recognition, ICDAR 2017, Kyoto, Japan, November 9–15, 2017, pp. 1192–1197. IEEE (2017). https://doi.org/10.1109/ICDAR.2017.197
15. Lample, G., Ballesteros, M., Subramanian, S., Kawakami, K., Dyer, C.: Neural architectures for named entity recognition. In: Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp. 260–270. Association for Computational Linguistics, San Diego (June 2016). https://www.aclweb.org/anthology/N16-1030
16. Martinek, J., Lenc, L., Král, P., Nicolaou, A., Christlein, V.: Hybrid training data for historical text OCR, pp. 565–570 (September 2019). https://doi.org/10.1109/ICDAR.2019.00096
17. Neudecker, C.: An open corpus for named entity recognition in historic newspapers. In: Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016), pp. 4348–4352. European Language Resources Association (ELRA), Portorož (May 2016). https://www.aclweb.org/anthology/L16-1689
18. Nothman, J., Ringland, N., Radford, W., Murphy, T., Curran, J.R.: Learning multilingual named entity recognition from Wikipedia. Artif. Intell. **194**, 151–175 (2013)
19. Piktus, A., Edizel, N.B., Bojanowski, P., Grave, E., Ferreira, R., Silvestri, F.: Misspelling oblivious word embeddings. In: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Long and Short Papers), vol. 1, pp. 3226–3234. Association for Computational Linguistics, Minneapolis (June 2019). https://www.aclweb.org/anthology/N19-1326
20. Ramponi, A., Plank, B.: Neural unsupervised domain adaptation in NLP–a survey (2020)
21. Schick, T., Schütze, H.: Rare words: a major problem for contextualized embeddings and how to fix it by attentive mimicking. CoRR abs/1904.06707 (2019). http://arxiv.org/abs/1904.06707
22. Schweter, S., Baiter, J.: Towards robust named entity recognition for historic German. In: Proceedings of the 4th Workshop on Representation Learning for NLP (RepL4NLP-2019), pp. 96–103. Association for Computational Linguistics, Florence (August 2019). https://www.aclweb.org/anthology/W19-4312
23. Schweter, S., März, L.: Triple E - effective ensembling of embeddings and language models for NER of historical German. In: Cappellato, L., Eickhoff, C., Ferro, N., Névéol, A. (eds.) Working Notes of CLEF 2020 - Conference and Labs of the Evaluation Forum, Thessaloniki, Greece, September 22–25, 2020. CEUR Workshop Proceedings, vol. 2696. CEUR-WS.org (2020). http://ceur-ws.org/Vol-2696/paper_173.pdf
24. Tjong Kim Sang, E.F.: Introduction to the CoNLL-2002 shared task: language-independent named entity recognition. In: COLING-02: The 6th Conference on Natural Language Learning 2002 (CoNLL-2002) (2002). https://www.aclweb.org/anthology/W02-2024
25. Tjong Kim Sang, E.F., De Meulder, F.: Introduction to the CoNLL-2003 shared task: language-independent named entity recognition. In: Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003, pp. 142–147 (2003). https://www.aclweb.org/anthology/W03-0419
26. Vobl, T., Gotscharek, A., Reffle, U., Ringlstetter, C., Schulz, K.U.: Pocoto - an open source system for efficient interactive postcorrection of ocred historical texts. In: Proceedings of the First International Conference on Digital Access to Textual

Cultural Heritage, DATeCH 2014, pp. 57–61. ACM, New York (2014). http://doi.acm.org/10.1145/2595188.2595197
27. Wagner, R.A., Fischer, M.J.: The string-to-string correction problem. J. ACM **21**(1), 168–173 (1974). https://doi.org/10.1145/321796.321811
28. Yeh, A.: More accurate tests for the statistical significance of result differences. In: The 18th International Conference on Computational Linguistics, COLING 2000, vol. 2 (2000). https://www.aclweb.org/anthology/C00-2137

4.3. hmBERT: Historical Multilingual Language Models for Named Entity Recognition



Contributing Article

Schweter, S., **März, L.**, Schmid, K., & Çano, E. (2022). hmBERT: Historical Multilingual Language Models for Named Entity Recognition. In *Proceedings of the Working Notes of CLEF 2022 - Conference and Labs of the Evaluation Forum, Bologna, Italy, September 5th - 8th, 2022.*
`http://ceur-ws.org/Vol-3180/paper-87.pdf`

Copyright Information

Author Contributions

The idea and conception for this work were brought up between Stefan Schweter and Luisa März and also discussed further jointly between all authors. Stefan Schweter contributed the pre-training algorithm for the historical language models and took over large parts of the implementation. He also put substantial effort into the transparent documentation and release of the models on the Hugging Face Model Hub. Luisa März supported and contributed constantly in the implementation of data-preprocessing, model training, and evaluation and mainly wrote the manuscript. Katharina Schmid contributed conceptual ideas and implemented the knowledge base experiment. Erion Çano was responsible for infrastructure and training and took care of the computational resources. The manuscript was constantly revised and proofread by all authors.

Supplementary Material

`https://github.com/dbmdz/clef-hipe/tree/main/experiments/clef-hipe-2022`

hmBERT: Historical Multilingual Language Models for Named Entity Recognition

Stefan Schweter[1], Luisa März[2,3], Katharina Schmid[1] and Erion Çano[2]

[1]Bayerische Staatsbibliothek München, Digital Library/ Munich Digitization Center, Munich, Germany
[2]Digital Philology, Research Group Data Mining and Machine Learning, University of Vienna, Austria
[3]Natural Language Processing Expert Center, Data:Lab, Volkswagen AG, Munich, Germany

Abstract

Compared to standard Named Entity Recognition (NER), identifying persons, locations, and organizations in historical texts constitutes a big challenge. To obtain machine-readable corpora, the historical text is usually scanned and Optical Character Recognition (OCR) needs to be performed. As a result, the historical corpora contain errors. Also, entities like location or organization can change over time, which poses another challenge. Overall, historical texts come with several peculiarities that differ greatly from modern texts and large labeled corpora for training a neural tagger are hardly available for this domain. In this work, we tackle NER for historical German, English, French, Swedish, and Finnish by training large historical language models. We circumvent the need for large amounts of labeled data by using unlabeled data for pretraining a language model. We propose hmBert, a historical multilingual BERT-based language model, and release the model in several versions of different sizes. Furthermore, we evaluate the capability of hmBert by solving downstream NER as part of this year's HIPE-2022 shared task and provide detailed analysis and insights. For the Multilingual Classical Commentary coarse-grained NER challenge, our tagger *HISTeria* outperforms the other teams' models for two out of three languages.

Keywords

Named Entity Recognition, historical NER, Transformer-based language models, Historical texts, Flair

1. Introduction

Standard Named Entity Recognition (NER) for high resource domains has already been successfully addressed with performances above 90% F1-score [1, 2]. In contrast, NER taggers often fail to achieve satisfying results in the historical domain. Since historical datasets usually stem from Optical Character Recognition (OCR) and also include domain shifts, they contain characteristic errors not found in modern text. Low-resource fonts like Fraktur pose additional challenges for clean OCR. Another problem is that large amounts of labeled data are required when training neural models and only little labeled data exists for historical NER [3]. Because of all these challenges, systems designed for contemporary datasets cannot be applied to the historical domain without adaptations or further training. However, in the last few years, a number of works have shown that it is possible to adapt systems by using different approaches [4].

CLEF 2022: Conference and Labs of the Evaluation Forum, September 5–8, 2022, Bologna, Italy
0000-0002-7190-2090 (S. Schweter); 0000-0003-4542-2437 (L. März); 0000-0001-6057-6640 (K. Schmid); 0000-0002-5496-3860 (E. Çano)

In this work, we develop a new BERT-based language model [5] for the historical context: hmBert. We tackle NER for historical German, English, French, Swedish, and Finnish. We use self-supervised learning to pretrain our language model on unlabeled data before we fine-tune the NER tagger on labeled data. This allows to reduce the need for large amounts of labeled training data. To mitigate the impact of OCR noise in the pretraining corpora, we use a filtering step that allows to control the OCR confidence of the texts.

Another design step in training language models is the choice of the underlying vocabulary. Beltagy et al. [6] showed that using a domain-specific vocabulary leads to performance improvements compared to using a general domain vocabulary. Thus, we use a sub-corpus of our pretraining corpus to create an in-domain vocabulary for the hmBert training. Finally, we arrive with a powerful hmBert model that establishes state-of-the-art results for three out of four languages on the NewsEye NER dataset [7]. As large language models require a lot of computational resources for pretraining and during inference time, we also provide smaller models.

Addressing the HIPE-2022 NERC-Coarse task, we also study a single-model vs. one-model approach. Our comparison shows that fine-tuning hmBert models for each language individually (single-model approach) improves performance compared to models that were fine-tuned on data from all languages (one-model approach). At the same time, however, fine-tuning individual models is much more computationally expensive. The one-model approach is more efficient, achieving similar performance while requiring less computation.

In addition, our final model *HISTeria* is trained by using multi-stage fine-tuning. We first fine-tune the multilingual model and evaluate it over the development data of all the different available languages. The resulting hyperparameter configuration is used for another fine-tuning step for each monolingual model. Finally, our detailed study of hmBert also includes experiments with a knowledge-based approach, training an ELECTRA-based language model [8], and addressing a tokenization issue. These additional experiments did not enhance performance but represent a suitable starting point for further research.

Our contributions are i) the comprehensive description of the development of hmBert, ii) the release of hmBert models of different sizes, iii) the release of the hmBert pretraining code, and iv) extensive experiments using hmBert including detailed insights for the community.

This paper is structured as follows: the next section (2) describes hmBert and its development in details. We include an explanation of the used datasets, as well as processing, hyperparameter settings, and pretraining steps. We close the section with a downstream task evaluation. Section 3 provides insights into the HIPE-2022 Multilingual Commentary Challenge[1] [9]. We describe our approach for the shared task submission in detail and provide an analysis of our results. We conclude the paper with Section 4.

2. hmBert: Historical Multilingual BERT Model

In this section, we present hmBert which supports German, English, French, Finnish, and Swedish. We train two different models with different vocabulary sizes: 32,000 and 64,000. We first describe the corpora used for training hmBert, as well as preprocessing and filtering

[1] https://hipe-eval.github.io/HIPE-2022/

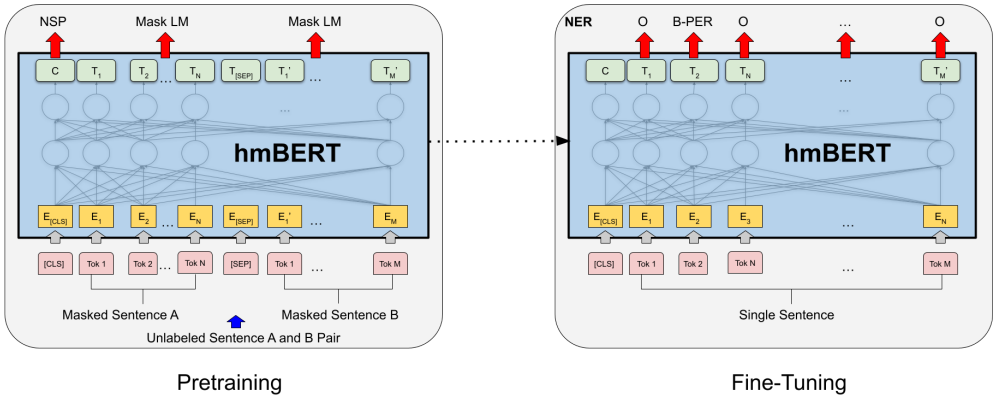


Figure 1: Overall pretraining of our $hmBERT_{32k}$ model and fine-tuning procedure for NER downstream tasks. CLS is a special symbol added in front of every input example, and SEP is a special separator token.

steps to create the pretraining corpus. In addition, we explain the pretraining process and end the section by evaluating the model on a downstream NER task. Figure 1 shows the overall pretraining procedure for hmBert and its application on downstream tasks.

2.1. Corpora

For German, French, Swedish and Finnish we use the Europeana newspapers[2] provided by the European Library. For English we use a dataset published by the British Library [10]. The dataset contains OCR-processed text from digitized books and has also been used by Hosseini et al. [11] to train historical language models for English.

2.1.1. Filtering

OCR full-text for the Europeana newspapers also includes an OCR confidence value. This measure indicates the average OCR confidence for each word of a newspaper[3]. For German and French we perform a number of characters per year analysis using different (minimum required) OCR confidence thresholds. For German, we test three different thresholds and report the resulting dataset size (see Table 15 in the appendix). We use an OCR confidence threshold of 0.60 to get a final dataset of approx. 28 GB. For French, we test five different OCR confidence values (see Table 16 in the appendix) and choose 0.70 so that the resulting dataset size of 27 GB is comparable to the size of the German dataset. For Finnish and Swedish, we use an OCR confidence threshold of 0.60. However, training data for Swedish and Finnish is very limited. In total, only 1.2 GB for Finnish, and 1.1 GB for Swedish are available, thus these corpora are not filtered any further using other OCR confidence thresholds. For English, language filtering using

[2] http://www.europeana-newspapers.eu/
[3] https://www.clarin.eu/sites/default/files/Nuno_Freire_Europeana_CLARINPLUS.pdf

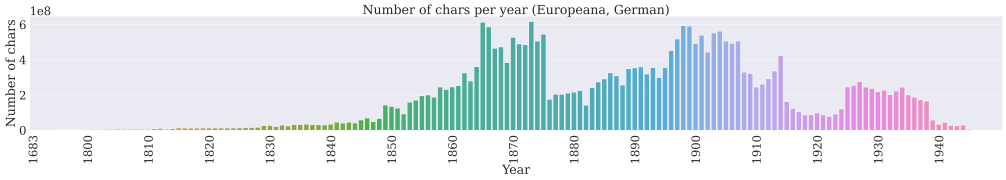


Figure 2: Number of characters per year distribution for filtered German Europeana corpus (1683-1949).

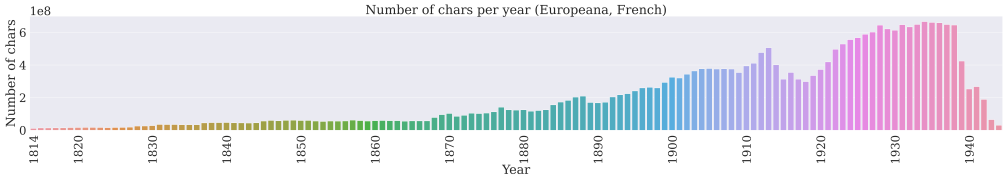


Figure 3: Number of characters per year distribution for filtered French Europeana corpus (1814-1944).

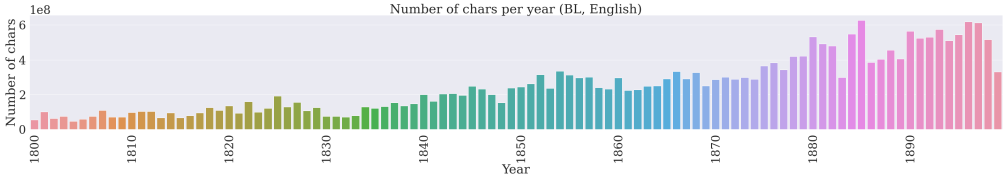


Figure 4: Number of characters per year distribution for filtered English corpus from British Library (1800-1899).

LANGDETECT[4] for each book in the corpus is performed. Additionally, we use books published between 1800 and 1900 exclusively. The resulting English corpus has a total size of 24 GB.

To get a deeper insight into the filtered corpora, we analyze the distribution of characters over time for each language. Figure 2 shows the distribution for German. The period from 1865 to 1914 is well-covered in the dataset, while the years from 1683 to 1849 and the 20th century are underrepresented. For French, the 20th century is highly covered, but there is only little data available for the 19th century (see Figure 3), which contrasts with the German corpus. The English corpus contains texts from the 19th century only and shows good coverage starting from 1850. However, there is only little coverage from 1800 to 1849 (see Figure 4). Since both Finnish and Swedish corpora include newspapers from 1900 to 1910 only, we do not analyze the number of characters per year for these datasets.

2.1.2. Multilingual Vocabulary Generation

To create a BERT-compatible wordpiece-based vocabulary [12], we use 10GB of each language and train the vocabulary using the Hugging Face Tokenizers library[5]. We build a cased vocabu-

[4] https://github.com/Mimino666/langdetect
[5] https://github.com/huggingface/tokenizers

Table 1
NER datasets that are used for calculating subword fertility rate and portion of UNKs. For English, the development dataset was used due to a missing training split.

Language	NER Corpora
German	CLEF-HIPE-2020 [15], NewsEye [7]
French	CLEF-HIPE-2020 [15], NewsEye [7]
English	CLEF-HIPE-2020 [15]
Finnish	NewsEye [7]
Swedish	NewsEye [7]

Table 2
Subword fertility rate and portion of UNKs calculated on NER datasets using a 32k wordpiece-based vocabulary.

Language	Subword Fertility	UNK Portion
German	1.43	0.0004
French	1.25	0.0001
English	1.25	0.0
Finnish	1.69	0.0007
Swedish	1.43	0.0

lary with no lower casing or accent stripping being performed. For Finnish and Swedish we need to upsample[6] the corpus because both corpora have a size of 1 GB only.

We create a 32k and 64k vocabulary. Inspired by Rust et al. [13], we report the subword fertility rate (SFR) and the portion of unknown (UNK) tokens per language on various historical NER datasets (see Table 1). The SFR is defined as the average number of subwords a tokenizer produces per word [13]. It indicates how aggressively a tokenizer splits, i.e. whether it over-segments or not. As over-segmentation can negatively impact downstream performance, an SFR close to 1 (indicating that the tokenizer vocabulary contains every word in the input text) is optimal. UNK tokens are challenging because such tokens are not seen during pretraining and the model cannot provide useful information for them during the fine-tuning phase [14]. Table 2 and Table 3 show the SFR and portion of UNKs in the 32k/64k corpus. French and English have the lowest SFRs, whereas Finnish has the highest rate in both wordpiece-based vocabularies.

2.2. Final Pretraining Corpus

For common multilingual models such as multilingual BERT [mBERT; 5], XLM-RoBERTa [16] or mT5 [17] different corpus sampling strategies have been developed to up-/downsample low-/high-resource languages [18]. Since our multilingual language model includes five languages only (mBERT covers 104 languages[7]), we use a similar size for all languages. After upsampling the Swedish and Finnish corpora to 27GB each, we arrive at a total dataset size of 130 GB. Table

[6]For upsampling we simply concatenate the original corpus N-times to match the desired 10 GB size per language.

[7]https://github.com/google-research/bert/blob/master/multilingual.md

Table 3
Subword fertility rate and portion of UNKs calculated on NER datasets using a 64k wordpiece-based vocabulary.

Language	Subword Fertility	UNK Portion
German	1.31	0.0004
French	1.16	0.0001
English	1.17	0.0
Finnish	1.54	0.0007
Swedish	1.32	0.0

Table 4
Size per language of final pretraining corpus for HMBERT.

Language	Dataset Size
German	28GB
French	27GB
English	24GB
Finnish	27GB
Swedish	27GB
Total	130GB

4 shows an overview of the sizes per language included in our final pretraining corpus. For the HMBERT model with a vocabulary size of 32k, we use the official BERT implementation[8] to create pretraining data. Detailed description of all parameters used for the creation of pretraining data can be found in Section A.2 of the appendix.

2.3. Models

We pretrain an HMBERT model with a vocabulary size of 32k, further denoted as $hmBERT_{32k}$, and another HMBERT model with a vocabulary size of 64k, further denoted as $hmBERT_{64k}$. Inspired by Hou et al. [19], we also pretrain and release smaller HMBERT models, with the number of layers ranging from 2 to 8 and hidden sizes ranging from 128 to 512. Pretraining of the different models is described in detail in Section A.3 of the appendix.

2.4. Downstream Task Evaluation

We evaluate the $hmBERT_{32k}$ models on the NewsEye NER dataset [7], because this dataset includes most of the languages that HMBERT covers (except English), and compare them with the current state-of-the-art reported by Hamdi et al. [20]. We use the FLAIR [21] library and perform a hyperparameter search (see Table 18 in appendix) using the common fine-tuning paradigm. Fine-tuning adds a single linear layer to a Transformer and fine-tunes the entire architecture on the NER downstream task. To bridge the difference between subword modeling

[8] https://github.com/google-research/bert#pre-training-with-bert

Table 5
Performance overview of $hmBERT_{32k}$ models on German NewsEye NER dataset.

Model Name	Development F1-Score	Test F1-Score
$hmBERT_{32k}$ Tiny	30.16	24.35
$hmBERT_{32k}$ Mini	35.74	31.54
$hmBERT_{32k}$ Small	40.27	39.04
$hmBERT_{32k}$ Medium	43.45	43,41
$hmBERT_{32k}$ Base	46.17	46.66
Hamdi et al. [20]	-	**48.3**

and token-level predictions, subword pooling is applied to create token-level presentations which are then passed to the final linear layer. A common subword pooling strategy is to use the first subtoken to represent the entire token and we also use this strategy in our experiments. To train our architecture, we use AdamW [22] optimizer, a very small learning rate and a fixed number of epochs as a hard-stopping criterion. We evaluate the model performance after each training epoch on the development set and use the best model (strict micro F1-score) for final evaluation. We adopt a one-cycle [23] training strategy, in which the learning rate linearly decreases until it reaches 0 by the end of the training. Tables 5 - 8 show the performance of our $hmBERT_{32k}$ models compared to the current state-of-the-art.

For German, even the $hmBERT_{32k}$ base model could not reach the performance reported by Hamdi et al. [20], that was based on the models developed by Boros et al. [24]. The performance difference is 1.64 percentage points. This could be due to the fact that the German NewsEye dataset is very large and the hyperparameter search needed to be extended. Furthermore, Hamdi et al. [20] proposed a new architecture for handling OCR errors by adding two extra transformer layers, whereas we only performed a standard fine-tuning approach. For French our $hmBERT_{32k}$ medium sized model is very close to the result reported by Hamdi et al. [20]. The $hmBERT_{32k}$ base model outperforms the current best result by +2.7 percentage points. The same performance gain can be observed for Finnish and Swedish: The $hmBERT_{32k}$ base model outperforms the current SOTA by 2.41 percentage points for Finnish, and 2.1 percentage points for the Swedish NewsEye dataset. Figure 5 shows an overall performance comparison for the pretrained $hmBERT_{32k}$ smaller models on the NewsEye dataset. On average, the performance difference between the 8-layer $hmBERT_{32k}$ medium and the 12-layer $hmBERT_{32k}$ base model is 2.7 percentage points.

3. HIPE-2022: Multilingual Classical Commentary Challenge

We participated in the Multilingual Classical Commentary Challenge (MCC) that was newly introduced in the 2022 edition of HIPE [25] with our tagger being denoted as *HISTeria*. The challenge requires participants to work with historical classical commentaries in at least two different languages and to develop solutions for Named Entity Recognition, Classification, and/or Linking. *HISTeria* aims to detect and classify named entities according to coarse-grained types (NERC-Coarse task) and is described in more detail in this section.

Table 6
Performance overview of $hmBERT_{32k}$ models on French NewsEye NER dataset.

Model Name	Development F1-Score	Test F1-Score
$hmBERT_{32k}$ Tiny	60.04	50.79
$hmBERT_{32k}$ Mini	70.55	62.28
$hmBERT_{32k}$ Small	75.72	69.02
$hmBERT_{32k}$ Medium	78.99	72.51
$hmBERT_{32k}$ Base	81.58	**75.10**
Hamdi et al. [20]	-	72.7

Table 7
Performance overview of $hmBERT_{32k}$ models on Finnish NewsEye NER dataset.

Model Name	Development F1-Score	Test F1-Score
$hmBERT_{32k}$ Tiny	30.37	34.76
$hmBERT_{32k}$ Mini	56.60	62.68
$hmBERT_{32k}$ Small	64.31	73.20
$hmBERT_{32k}$ Medium	69.95	76.34
$hmBERT_{32k}$ Base	76.05	**80.11**
Hamdi et al. [20]	-	77.7

Table 8
Performance overview of $hmBERT_{32k}$ models on Swedish NewsEye NER dataset.

Model Name	Development F1-Score	Test F1-Score
$hmBERT_{32k}$ Tiny	43.65	38.91
$hmBERT_{32k}$ Mini	64.05	65.58
$hmBERT_{32k}$ Small	73.47	76.29
$hmBERT_{32k}$ Medium	78.07	82.47
$hmBERT_{32k}$ Base	81.13	**83.60**
Hamdi et al. [20]	-	81.5

3.1. Data

A classical commentary is a scholarly publication that aims to facilitate the reading and understanding of classical works of literature by providing additional information such as translations or bibliographic references. Apart from the challenges that are common to historical texts, commentaries have other characteristics that may complicate Named Entity Recognition and Classification: they frequently cite the original literary text, making them inherently multilingual, and they often use abbreviations to convey information more concisely. For the Multilingual Classical Commentary Challenge, HIPE[9] has chosen a single dataset that was

[9] https://github.com/hipe-eval/HIPE-2022-data/blob/main/documentation/README-ajmc.md

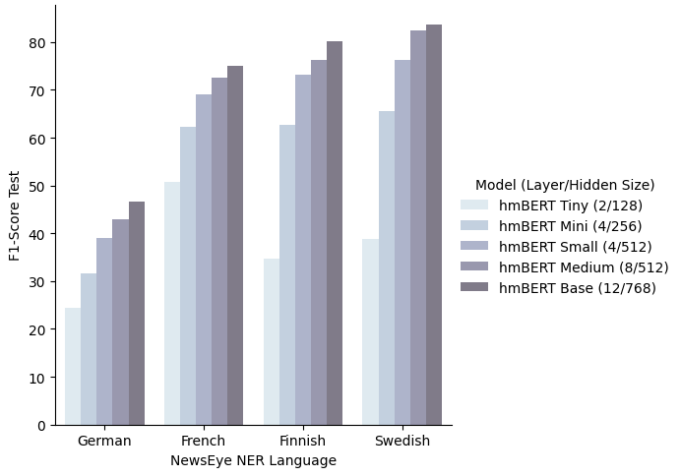


Figure 5: Overview of performance of $hmBERT_{32k}$ smaller models on NewsEye NER datasets. F1-score on the test set is reported here.

Table 9
Dataset statistics about `ajmc` dataset.

Language	Training Sentences	Development Sentences
German	1,024	192
English	1,154	252
French	894	202

created in the context of the Ajax MultiCommentary project[10] (`ajmc` dataset). The dataset contains excerpts from commentaries published in the 19th century in English, French, and German. The French texts date from 1886, the German ones from 1853 and 1894, and the English ones from 1881 and 1896. This emphasis on the second half of the 19th century fits well with the temporal distribution of our pretraining data for English and German. Apart from standard entity types like *person* or *location*, the dataset also includes domain-specific annotations like the *scope* and *work* entity type for bibliographic references. Additional dataset statistics can be found in Table 9 and in the HIPE 2022 Overview paper [9].

3.2. Single-Models vs. One-Model Approach

In preliminary experiments models that are independently fine-tuned for each language (single-model approach) and a model that uses training data from all languages (one-model approach) are compared. We perform hyperparameter searches for the two approaches. The relevant hyperparameters for fine-tuning models are shown in the appendix (Table 19). We use the Flair library for all experiments. For the one-model approach, a breakdown analysis for each language

[10] https://mromanello.github.io/ajax-multi-commentary/

Table 10
Performance comparison for NERC-coarse between single-model and one-model approach on ajmc development dataset. Numbers express F1-score calculated by using the strict evaluation regime.

Language	Single-Model	One-Model
German	86.21	**86.68**
English	**84.98**	84.85
French	**85.69**	85.09

is performed after determining the best hyperparameter configuration. This is compared to the three independently fine-tuned models for each language. For German, the one-model approach is +0.47 percentage points better than the single-model approach. For English, the one-model approach performs slightly worse (-0.13 percentage points) and for French, the single-model approach outperforms the one-model by 0.6 percentage points. However, the single-model approach requires fine-tuning of 120 models, whereas the one-model approach only needs 40 models to be fine-tuned for hyperparameter search. To save resources, we decided to use the one-model approach for further experiments. The performance comparison on the ajmc dataset is shown in Table 10.

3.3. Multi-Stage Fine-Tuning

Wang et al. [26] proposed a knowledge-based system for multilingual NER using a multi-stage fine-tuning approach for the MultiCoNER SemEval 2022 task[11]. The first stage of multi-stage fine-tuning refers to training a multilingual model on data from different languages. In the second stage, this fine-tuned multilingual model is used as a starting point for training a monolingual model. We adapt this approach for our final system: in the first stage, we fine-tune one multilingual model over the training data of all three languages (German, English, and French) and optimize over all development data (one-model approach) using a hyperparameter search. We select the best hyperparameter configuration as a combination of batch size, the number of epochs, and the learning rate, which results in five models (because of five different random seeds). The hyperparameter search grid for the different stages is shown in Section B.1 in the appendix. From these five models, we choose the one with the highest F1-score on the development set for second stage fine-tunings. In the second stage, we use the best model from the first stage and fine-tune single models for each language with a hyperparameter search on the development set. For each language, we select the best hyperparameter configuration and choose the best performing model with the highest F1-score on the development set. In preliminary experiments, this multi-stage fine-tuning approach boosts performance by 1.23 percentage points on average compared to results in the first stage.

For our final submission, $hmBERT_{32k}$ achieves the best results during the first-stage of fine-tuning with a batch size of 4, 10 fine-tuning epochs and a learning rate of $5e-05$. This results in an average F1-score of 86.89 on the (combined) development sets for ajmc. The best hyperparameter configuration for $hmBERT_{64k}$ can be achieved when using a batch size of 8,

[11] https://multiconer.github.io/

Table 11
Final results on ajmc development dataset for all languages using best models after multi-stage fine-tuning. Results are reported with official HIPE scorer.

Submission ID	Hyperparameter Configuration	Strict F1-Score	Fuzzy F1-Score
German ($hmBERT_{32k}$) - 1	bs8-e05-lr3e-05	91.5	**94.2**
German ($hmBERT_{64k}$) - 2	bs8-e10-lr3e-05	**92.0**	93.9
English ($hmBERT_{32k}$) - 1	bs4-e10-lr3e-05	**89.1**	92.9
English ($hmBERT_{64k}$) - 2	bs8-e10-lr3e-05	88.0	**93.8**
French ($hmBERT_{32k}$) - 1	bs4-e10-lr3e-05	**86.8**	**93.1**
French ($hmBERT_{64k}$) - 2	bs4-e10-lr5e-05	85.9	93.0

10 epochs of fine-tuning and a learning rate of $3e-05$. This results in an overall F1-score of 86.69 percentage points. Thus, $hmBERT_{64k}$ is slightly worse than $hmBERT_{32k}$ (-0.2 percentage points). Table 11 shows the performance for our final submissions using $hmBERT_{32k}$ and $hmBERT_{64k}$ for all languages in the ajmc dataset. We report strict and fuzzy F1-scores using the official HIPE-scorer[12]. We exclude document-level scores for better readability.

3.4. *HISTeria* Results

Table 12 shows an overview of *HISTeria* compared to the runs of other teams in the HIPE-2022 shared task[13] .

To gain a better understanding of our models, we use the attribute-aided evaluation proposed by Fu et al. [27]. In order to highlight the strengths and weaknesses of different models, they analyze how model performance varies with regard to certain attributes. In the case of NER, properties that may influence performance are i) how consistently a given surface form of a token or an entity is labelled across a dataset (tCon and eCon), ii) how often a given token or entity appears in the dataset (tFre and eFre), iii) the number of tokens that make up an entity (eLen) or sentence (sLen) as well as iv) the relative number of out-of-vocabulary words and entities per sentence (oDen and eDen). Using the implementation by Fu et al. [27], we distribute the values into buckets and compute the strict F1-score for each bucket. Table 13 shows Spearman's rank correlation coefficient as a measure of how well the attribute correlates with the F1-score, and the standard deviation of the F1-score to indicate how strongly the attribute influences performance. We omit results that are not statistically significant.

For the two German models, none of the attributes seem to correlate with performance in a statistically significant way. For the English and French models, performance correlates directly and positively with the consistency of the token labels. The standard deviation of 10% (French) and 8-9% (English) of the F1-score indicates that this attribute has a marked impact on performance. For French $hmBERT_{32k}$, entity length influences performance to the same degree. In this case, performance gets worse the more tokens an entity has. The impact of entity length on English $hmBERT_{32k}$ and $hmBERT_{64k}$ is less strong but still notable (standard

[12]https://github.com/hipe-eval/HIPE-scorer
[13]https://github.com/hipe-eval/HIPE-2022-eval/

Table 12
Final results on ajmc test dataset for all languages compared to other participants in the HIPE-2022 shared task. HISTeria denotes our system. Rank is ordered by strict F1-score.

Rank	Language	Submission ID	Strict F1-Score	Fuzzy F1-Score
1	German	L3i (team 2) - 2	93.4	95.2
2	German	HISTeria ($hmBERT_{32k}$) - 1	91.3	93.7
3	German	HISTeria ($hmBERT_{64k}$) - 2	91.2	94.5
4	German	L3i (team 2) - 1	90.8	93.4
5	German	Neural baseline	81.8	87.3
1	English	HISTeria ($hmBERT_{64k}$) - 2	85.4	91.0
2	English	L3i (team 2) - 1	85.0	89.4
3	English	L3i (team 2) - 2	84.1	88.4
4	English	HISTeria ($hmBERT_{32k}$) - 1	81.9	89.9
5	English	Neural baseline	73.6	82.8
1	French	HISTeria ($hmBERT_{64k}$) - 2	84.2	88.0
2	French	HISTeria ($hmBERT_{32k}$) - 1	83.3	88.8
3	French	L3i (team 2) - 2	82.6	87.2
4	French	L3i (team 2) - 1	79.8	86.0
5	French	Neural baseline	74.1	82.5

Table 13
Spearman's rank correlation coefficient and standard deviation of models' F1-score depending on different attribute values. We omit results that are not statistically significant.

Model	Attribute	Spearman	Standard Deviation
English $hmBERT_{32k}$	tCon	1.0	0.09
	eLen	-1.0	0.06
	oDen	-1.0	0.09
English $hmBERT_{64k}$	tCon	1.0	0.08
	eLen	-1.0	0.01
French $hmBERT_{32k}$	tCon	1.0	0.10
	eLen	-1.0	0.10
French $hmBERT_{64k}$	tCon	1.0	0.10

deviation of 6% and 1% respectively). In addition to entity length, the amount of words that did not feature in the training set also correlates negatively with the performance of English $hmBERT_{32k}$.

3.5. Challenges

We also experimented with the knowledge-based system for multilingual NER that was proposed by Wang et al. [26]. We used their implementation to enrich the original ajmc datasets with a knowledge base and implemented their context approach in the Flair library. More precisely,

we used the FLERT approach [28] and utilized the knowledge-base enriched context as the left context for each training example. A left context size of 128 performs best in the experiments. However, the final result was slightly worse than using no context at all. This may be due to the fact that a contemporary, general-purpose knowledge base (Wikipedia) was used. A domain-specific knowledge base may yield better results. As the preliminary results were slightly worse than our main baseline, we did not conduct further experiments with this knowledge-based system.

We calculated the portion of UNKs in the German ajmc dataset and found that the portion rate of 16.3 % is unreasonably high. We discovered that the German ajmc dataset contains long-s characters, unlike the Europeana Newspaper corpora which were used to train a vocabulary. As a consequence, the hmBert tokenizer is not able to handle tokens that include long-s characters, resulting in UNKs. For our final system, we manually replaced all long-s characters with a normal s character to circumvent the UNK problem. In upcoming versions of our hmBert models, we will add this replacement step in the tokenizer configuration directly. Furthermore, we also trained an ELECTRA model [8] for 1M steps on the same pretraining corpus as the $hmBERT_{32k}$ model. We found that the downstream performance on NewsEye datasets was 1 to 3 percentage points worse than $hmBERT_{32k}$ and -0.28 percentage points worse on the ajmc dataset. We have therefore decided not to release the model yet.

3.6. Community Contributions

To foster research on language and NER models for the historical domain, we publicly release our pretrained and fine-tuned models on the Hugging Face Model Hub[14] under the dbmdz namespace[15]. We also publicly release all code that was used for fine-tuning models[16]. Table 14 shows an overview of released models for our HIPE-2022 submission, including the model identifier on the Hugging Face Model Hub. All models are released under a permissive MIT license. Additionally, we added dataset support for all HIPE-2022 NER datasets into Flair library[17].

4. Conclusion

We presented hmBert, a new multilingual BERT-based language model for historical data. hmBert is composed of German, French, English, Finnish, and Swedish unsupervised corpora of historical OCR-processed texts. The corpora have been filtered for OCR confidence as well as sampled so that each language contributes a similar amount of data to the model. The underlying vocabulary is also derived from each of the languages used for hmBert. In our temporal analysis of the pretraining corpora, we have found that data from the 18th and 19th century is unevenly distributed across the different languages. For future models, we are looking for additional datasets to balance this representation. We evaluated two hmBert models of different sizes with downstream Named Entity Recognition. For the NewsEye dataset hmBert

[14] https://huggingface.co/
[15] https://huggingface.co/dbmdz
[16] https://github.com/dbmdz/clef-hipe
[17] Added in Flair version 0.11: https://github.com/flairNLP/flair/releases/tag/v0.11

Table 14
Community contributions for our HIPE-2022 submission: Pretrained language models and fine-tuned NER models are publicly available on the Hugging Face Model Hub.

Model Description	Model Name
$hmBERT_{32k}$ Tiny Model	dbmdz/bert-tiny-historic-multilingual-cased
$hmBERT_{32k}$ Mini Model	dbmdz/bert-mini-historic-multilingual-cased
$hmBERT_{32k}$ Small Model	dbmdz/bert-small-historic-multilingual-cased
$hmBERT_{32k}$ Medium Model	dbmdz/bert-medium-historic-multilingual-cased
$hmBERT_{32k}$ Base Model	dbmdz/bert-base-historic-multilingual-cased
$hmBERT_{64k}$ Base Model	dbmdz/bert-base-historic-multilingual-64k-td-cased
NER First Stage ($hmBERT_{32k}$)	dbmdz/flair-hipe-2022-ajmc-all
NER First Stage ($hmBERT_{64k}$)	dbmdz/flair-hipe-2022-ajmc-all-64k
NER Second Stage - German ($hmBERT_{32k}$)	dbmdz/flair-hipe-2022-ajmc-de
NER Second Stage - English ($hmBERT_{32k}$)	dbmdz/flair-hipe-2022-ajmc-en
NER Second Stage - French ($hmBERT_{32k}$)	dbmdz/flair-hipe-2022-ajmc-fr
NER Second Stage - German ($hmBERT_{64k}$)	dbmdz/flair-hipe-2022-ajmc-de-64k
NER Second Stage - English ($hmBERT_{64k}$)	dbmdz/flair-hipe-2022-ajmc-en-64k
NER Second Stage - French ($hmBERT_{64k}$)	dbmdz/flair-hipe-2022-ajmc-fr-64k

established a new state-of-the-art for three out of four languages: French, Finnish, and Swedish. For the 2022 HIPE Multilingual Classical Commentary Challenge, our *HISTeria* system could outperform the other systems for two out of three languages. Using multi-stage fine-tuning together with the multilingual BERT-based model led the model to its optimal performance. Detailed analysis showed the benefits of all of hmBERTs design choices, as well as interesting findings for future research. Our contributions include all of the trained hmBERT models and our source code, which are made publicly available.

Acknowledgments

We would like to thank Google's TPU Research Cloud (TRC) program for giving us access to TPUs that were used for training our hmBERT models. We would also like to thank Hugging Face for providing the ability to host and perform inferencing of our models on the Hugging Face Model Hub.

References

[1] A. Akbik, T. Bergmann, R. Vollgraf, Pooled Contextualized Embeddings for Named Entity Recognition, in: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), Association for Computational Linguistics, Minneapolis, Minnesota, 2019, pp. 724–728. doi:10.18653/v1/N19-1078.

[2] X. Wang, Y. Jiang, N. Bach, T. Wang, Z. Huang, F. Huang, K. Tu, Automated Concatenation of Embeddings for Structured Prediction, in: the Joint Conference of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International

Joint Conference on Natural Language Processing (ACL-IJCNLP 2021), Association for Computational Linguistics, 2021.

[3] M. Ehrmann, G. Colavizza, Y. Rochat, F. Kaplan, Diachronic Evaluation of NER Systems on Old Newspapers, Bochumer Linguistische Arbeitsberichte, Bochum, Germany, 2016, pp. 97–107. URL: http://infoscience.epfl.ch/record/221391.

[4] M. Ehrmann, A. Hamdi, E. Linhares Pontes, M. Romanello, A. Douvet, A Survey of Named Entity Recognition and Classification in Historical Documents, ACM Computing Surveys (2022 (to appear)). URL: https://arxiv.org/abs/2109.11406.

[5] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, in: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), Association for Computational Linguistics, Minneapolis, Minnesota, 2019, pp. 4171–4186. doi:10.18653/v1/N19-1423.

[6] I. Beltagy, K. Lo, A. Cohan, SciBERT: A Pretrained Language Model for Scientific Text, in: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), Association for Computational Linguistics, Hong Kong, China, 2019, pp. 3615–3620. doi:10.18653/v1/D19-1371.

[7] A. Hamdi, E. L. Pontes, E. Boros, T. T. H. Nguyen, G. Hackl, J. G. Moreno, A. Doucet, Multilingual Dataset for Named Entity Recognition, Entity Linking and Stance Detection in Historical Newspapers, 2021. doi:10.5281/zenodo.4573313.

[8] K. Clark, M.-T. Luong, Q. V. Le, C. D. Manning, ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators, in: International Conference on Learning Representations, 2020. URL: https://openreview.net/forum?id=r1xMH1BtvB.

[9] M. Ehrmann, M. Romanello, S. Najem-Meyer, A. Doucet, S. Clematide, Overview of HIPE-2022: Named Entity Recognition and Linking in Multilingual Historical Documents, in: A. Barrón-Cedeño, G. Da San Martino, M. Degli Esposti, F. Sebastiani, C. Macdonald, G. Pasi, A. Hanbury, M. Potthast, G. Faggioli, N. Ferro (Eds.), Experimental IR Meets Multilinguality, Multimodality, and Interaction. Proceedings of the Thirteenth International Conference of the CLEF Association (CLEF 2022), Lecture Notes in Computer Science (LNCS), Springer, 2022.

[10] B. L. Labs, Digitised books. c. 1510 - c. 1946. json (ocr derived text)., 2016. doi:10.21250/DB14.

[11] K. Hosseini, K. Beelen, G. Colavizza, M. Coll Ardanuy, Neural Language Models for Nineteenth-Century English, arXiv e-prints (2021) arXiv:2105.11321. arXiv:2105.11321.

[12] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, J. Klingner, A. Shah, M. Johnson, X. Liu, L. Kaiser, S. Gouws, Y. Kato, T. Kudo, H. Kazawa, K. Stevens, G. Kurian, N. Patil, W. Wang, C. Young, J. Smith, J. Riesa, A. Rudnick, O. Vinyals, G. Corrado, M. Hughes, J. Dean, Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation, CoRR abs/1609.08144 (2016). URL: http://arxiv.org/abs/1609.08144. arXiv:1609.08144.

[13] P. Rust, J. Pfeiffer, I. Vulić, S. Ruder, I. Gurevych, How Good is Your Tokenizer? On the Monolingual Performance of Multilingual Language Models, in: Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th

International Joint Conference on Natural Language Processing (Volume 1: Long Papers), Association for Computational Linguistics, Online, 2021, pp. 3118–3135. doi:10.18653/v1/2021.acl-long.243.

[14] J. Pfeiffer, I. Vulić, I. Gurevych, S. Ruder, UNKs Everywhere: Adapting Multilingual Language Models to New Scripts, in: Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Online and Punta Cana, Dominican Republic, 2021, pp. 10186–10203. doi:10.18653/v1/2021.emnlp-main.800.

[15] M. Ehrmann, M. Romanello, A. Flückiger, S. Clematide, Extended Overview of CLEF HIPE 2020: Named Entity Processing on Historical Newspapers, volume 2696 of *CEUR Workshop Proceedings. 2696*, CEUR-WS, 2020, p. 38. doi:10.5281/zenodo.4117566.

[16] A. Conneau, K. Khandelwal, N. Goyal, V. Chaudhary, G. Wenzek, F. Guzmán, E. Grave, M. Ott, L. Zettlemoyer, V. Stoyanov, Unsupervised Cross-lingual Representation Learning at Scale, in: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, Association for Computational Linguistics, Online, 2020, pp. 8440–8451. doi:10.18653/v1/2020.acl-main.747.

[17] L. Xue, N. Constant, A. Roberts, M. Kale, R. Al-Rfou, A. Siddhant, A. Barua, C. Raffel, mT5: A Massively Multilingual Pre-trained Text-to-Text Transformer, in: Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Association for Computational Linguistics, Online, 2021, pp. 483–498. doi:10.18653/v1/2021.naacl-main.41.

[18] A. Conneau, G. Lample, Cross-lingual Language Model Pretraining, in: H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, R. Garnett (Eds.), Advances in Neural Information Processing Systems, volume 32, Curran Associates, Inc., 2019. URL: https://proceedings.neurips.cc/paper/2019/file/c04c19c2c2474dbf5f7ac4372c5b9af1-Paper.pdf.

[19] L. Hou, R. Yuanzhe Pang, T. Zhou, Y. Wu, X. Song, X. Song, D. Zhou, Token Dropping for Efficient BERT Pretraining, arXiv e-prints (2022) arXiv:2203.13240. arXiv:2203.13240.

[20] A. Hamdi, E. Boroş, E. L. Pontes, T. T. H. Nguyen, G. Hackl, J. G. Moreno, A. Doucet, A Multilingual Dataset for Named Entity Recognition, Entity Linking and Stance Detection in Historical Newspapers, in: Proceedings of the 44rd International ACM SIGIR Conference on Research and Development in Information Retrieval, 2021.

[21] A. Akbik, T. Bergmann, D. Blythe, K. Rasul, S. Schweter, R. Vollgraf, FLAIR: An easy-to-use framework for state-of-the-art NLP, in: NAACL 2019, 2019 Annual Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations), 2019, pp. 54–59.

[22] I. Loshchilov, F. Hutter, Decoupled weight decay regularization, in: International Conference on Learning Representations, 2019. URL: https://openreview.net/forum?id=Bkg6RiCqY7.

[23] L. N. Smith, A disciplined approach to neural network hyper-parameters: Part 1 – learning rate, batch size, momentum, and weight decay, arXiv e-prints (2018) arXiv:1803.09820. arXiv:1803.09820.

[24] E. Boros, A. Hamdi, E. Linhares Pontes, L. A. Cabrera-Diego, J. G. Moreno, N. Sidere, A. Doucet, Alleviating Digitization Errors in Named Entity Recognition for Historical Documents, in: Proc. of the 24th Conference on Computational Natural Language Learning,

ACL, 2020, pp. 431–441. URL: https://www.aclweb.org/anthology/2020.conll-1.35.

[25] M. Ehrmann, M. Romanello, S. Clematide, A. Doucet, Introducing the HIPE 2022 Shared Task:Named Entity Recognition and Linking in Multilingual Historical Documents, in: Proceedings of the 44^{d} European Conference on IR Research (ECIR 2022), Lecture Notes in Computer Science, Springer, Stavanger, Norway, 2022. URL: https://link.springer.com/chapter/10.1007/978-3-030-99739-7_44.

[26] X. Wang, Y. Shen, J. Cai, T. Wang, X. Wang, P. Xie, F. Huang, W. Lu, Y. Zhuang, K. Tu, W. Lu, Y. Jiang, DAMO-NLP at SemEval-2022 Task 11: A Knowledge-based System for Multilingual Named Entity Recognition (2022). URL: https://arxiv.org/abs/2112.06482. arXiv:2203.00545.

[27] J. Fu, P. Liu, G. Neubig, Interpretable Multi-dataset Evaluation for Named Entity Recognition, in: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), Association for Computational Linguistics, Online, 2020, pp. 6058–6069. doi:10.18653/v1/2020.emnlp-main.489.

[28] S. Schweter, A. Akbik, FLERT: Document-Level Features for Named Entity Recognition, 2020. arXiv:2011.06993.

[29] S. Schweter, BERTurk - BERT models for Turkish, 2020. doi:10.5281/zenodo.3770924.

[30] L. Hou, R. Y. Pang, T. Zhou, Y. Wu, X. Song, X. Song, D. Zhou, Token Dropping for Efficient BERT Pretraining, in: Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Association for Computational Linguistics, Dublin, Ireland, 2022, pp. 3774–3784. URL: https://aclanthology.org/2022.acl-long.262.

A. hmBERT: Historical Multilingual BERT Model

A.1. Corpora Filtering

Table 15
Word level OCR confidence thresholds for German. Bold OCR confidence is used for the final corpus.

OCR Confidence	Dataset Size
0.60	28GB
0.65	18GB
0.70	13GB

Table 16
Word level OCR confidence thresholds for French. Bold OCR confidence is used for the final corpus.

OCR Confidence	Dataset Size
0.60	31GB
0.65	27GB
0.70	27GB
0.75	23GB
0.80	11GB

A.2. Final Pretraining Corpus

For creation of the pretraining data, we use the same parameters as BERTurk [29]: maximum sequence length = 512, maximum predictions per sequence = 75, masked language probability rate = 0.15, duplication factor = 5. Due to hardware limitations, we split the pretraining corpus into chunks of 1GB and create pretraining data for each chunk individually. For the hmBert model with a vocabulary size of 64k we use the official implementation[18] with the same parameters as for the 32k model, but we increase the maximum predictions per sequence to 76.

A.3. Models

We use the official BERT implementation[19] for pretraining $hmBERT_{32k}$. $hmBERT_{64k}$ is trained with the recently proposed "token dropping" approach by Hou et al. [30]. Using this approach, unimportant tokens starting from an intermediate layer in the model are dropped to make the model focus on important tokens more efficiently, which makes model pretraining faster compared to the original BERT implementation. For both pretraining approaches, we use a maximum sequence length of 512 for the full training time. For the pretraining of $hmBERT_{32k}$ a batch size of 128 is used for 3M training steps. Pretraining was done on a v3-32 TPU pod

[18] https://github.com/tensorflow/models/blob/27fb855b027ead16d2616dcb59c67409a2176b7f/official/legacy/bert/README.md#pre-training

[19] https://github.com/google-research/bert

within 67 hours. The pretraining of $hmBERT_{64k}$ was done on a single v4-8 TPU with a batch size of 512 for 1M steps within 114 hours. Figure 6 shows the pretraining loss for $hmBERT_{32k}$ and $hmBERT_{64k}$. The final $hmBERT_{32k}$ has 110.62M, whereas $hmBERT_{64k}$ has 135.19M parameters due to the increased vocabulary size.

For better comparability, we measure the number of total subtokens seen during pretraining[20] and the number of total subtokens of the pretraining corpus for our two HMBERT models. More precisely, $hmBERT_{32k}$ has seen 196B subtokens during pretraining, whereas the pretraining corpus has a total size of 42B subtokens. This results in 4.7 pretraining epochs over the corpus. Our $hmBERT_{64k}$ model has seen 262B subtokens during pretraining. Because of the larger vocabulary size, the number of subtokens for the corpus is 39B. This results in 6.7 pretraining epochs over the corpus.

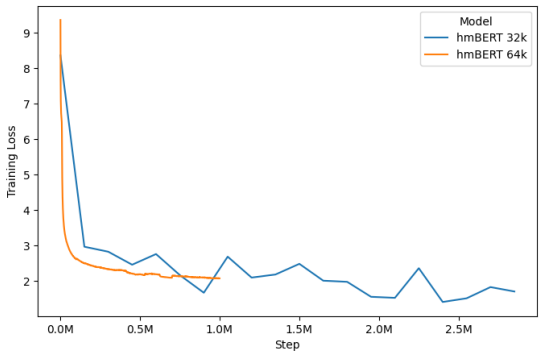


Figure 6: Overview of pretraining loss for $hmBERT_{32k}$ and $hmBERT_{64k}$.

For the smaller models, we use the same pretraining data and hyperparameter as for the base $hmBERT_{32k}$ model and pretrain them on a v3-32 TPU pod. Table 17 shows an overview of pretrained models, including their model size, number of parameters and pretraining time. Figure 7 shows an overview of pretraining loss for all smaller $hmBERT_{32k}$ models.

Table 17
Overview of smaller $hmBERT_{32k}$ models with their corresponding model size, number of parameters and pretraining time.

Model Name	Number of Layers	Hidden Size	Parameters	Pretraining Time
$hmBERT_{32k}$ Tiny	2	128	4.58M	4.3s / 1k steps
$hmBERT_{32k}$ Mini	4	256	11.55M	10.5s / 1k steps
$hmBERT_{32k}$ Small	4	512	29.52M	20.7s / 1k steps
$hmBERT_{32k}$ Medium	8	512	42.13M	35.0s / 1k steps
$hmBERT_{32k}$ Base	12	768	110.62M	80.0s / 1k steps

[20] Total number of subtokens during pretraining can be calculated as multiplication of training steps, batch size and sequence length

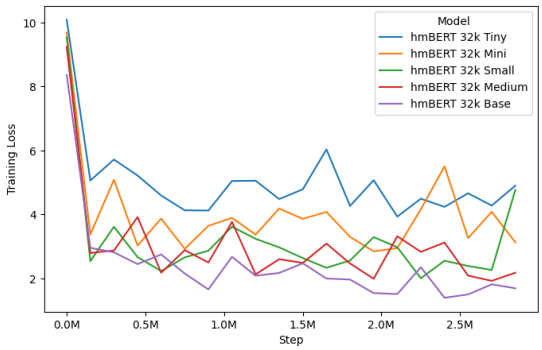


Figure 7: Overview of pretraining loss for smaller $hmBERT_{32k}$ models.

A.4. Downstream Task Evaluation

Table 18
Hyperparameter search for downstream evaluation on NewsEye NER dataset.

Parameter	Values
Batch Size	$[4, 8]$
Epoch	$[5, 10]$
Learning Rate	$[3e-05, 5e-05]$
Seed	$[1, 2, 4, 5]$

B. HIPE-2022: Multilingual Classical Commentary Challenge

B.1. Multi-Stage Fine-Tuning

Table 19
Hyperparameter search during the first stage of NER model fine-tuning.

Parameter	Values
Batch Size	$[4, 8, 16]$
Epoch	$[10]$
Learning Rate	$[1e-05, 2e-05, 3e-05, 4e-05, 5e-05]$
Seed	$[1, 2, 4, 5]$

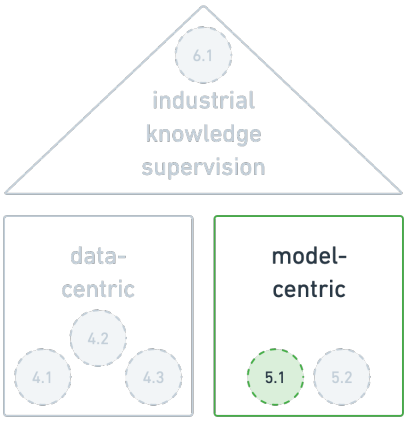
Table 20
Hyperparameter search during the second stage of NER model fine-tuning.

Parameter	Values
Batch Size	$[4, 8]$
Epoch	$[5, 10]$
Learning Rate	$[3e-05, 5e-05]$
Seed	$[1, 2, 4, 5]$

As a batch size of 16 and learning rates of $1e-05$ and $2e-05$ do not perform well, we exclude them when performing hyperparameter search with $hmBERT_{64k}$.

5. Model-Centric Knowledge Supervision

5.1. KnowMAN: Weakly Supervised Multinomial Adversarial Networks



Contributing Article

März, L., Asgari, E., Braune, F., Zimmermann, F., & Roth, B. (2021). KnowMAN: Weakly Supervised Multinomial Adversarial Networks. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 9549–9557, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics. `https://aclanthology.org/2021.emnlp-main.751`

Copyright Information

Author Contributions

The method and conception of the work were brought up in discussions between Luisa März and Benjamin Roth. Luisa März did the majority of the programming and writing, with constant revision from the other authors. The parts Luisa März programmed include data compilation, data pre-processing, PyTorch-based model implementation and training, as well as scripts for prediction, evaluation, and analysis. Ehsan Asgari, Fabienne Braune, and Franziska Zimmermann contributed with constant feedback and discussion on the method and took over smaller parts of the implementation. Luisa März and Benjamin Roth finally revised and proofread the manuscript together.

Supplementary Material

Code: `https://github.com/LuisaMaerz/KnowMAN`

KnowMAN: Weakly Supervised Multinomial Adversarial Networks

Luisa März [◇,†], **Ehsaneddin Asgari** [†], **Fabienne Braune** [†],
Franziska Zimmermann[†] **and Benjamin Roth** [◇]
[◇] Digital Philology, Research Group Data Mining and Machine Learning,
University of Vienna, Austria
[†] NLP Expert Center, Data:Lab, Volkswagen AG, Munich, Germany

Abstract

The absence of labeled data for training neural models is often addressed by leveraging knowledge about the specific task, resulting in heuristic but noisy labels. The knowledge is captured in labeling functions, which detect certain regularities or patterns in the training samples and annotate corresponding labels for training. This process of weakly supervised training may result in an over-reliance on the signals captured by the labeling functions and hinder models to exploit other signals or to generalize well. We propose KnowMAN, an adversarial scheme that enables to control influence of signals associated with specific labeling functions. KnowMAN forces the network to learn representations that are invariant to those signals and to pick up other signals that are more generally associated with an output label. KnowMAN strongly improves results compared to direct weakly supervised learning with a pre-trained transformer language model and a feature-based baseline.

1 Introduction

Neural approaches rely on labeled data sets for training. For many tasks and languages, such data is either scarce or not available at all. Knowledge-based weak supervision tackles this problem by employing *labeling functions (LFs)*. LFs are manually specified properties, e.g. keywords, that trigger the automatic annotation of a specific label. However, these annotations contain noise and biases that need to be handled.

A recent approach for denoising weakly supervised data is Snorkel (Ratner et al., 2020). Snorkel focuses on estimating the reliability of LFs and of the resulting heuristic *labels*. However, Snorkel does not address biases on the *input side* of weakly supervised data, which might lead to learned representations that overfit the characteristics of specific LFs, hindering generalization. We address the problem of overfitting to the LFs in this paper.

Other approaches tackle such overfitting by deleting the LF signal completely from the input side of an annotated sample: For example, Go et al. (2009) strip out emoticons that were used for labeling the sentiment in tweets, and Alt et al. (2019) mask the entities used for distant supervision of relation extraction training data (Mintz et al., 2009). However, as LFs are often constructed from the most prototypical and reliable signals (e.g., keywords), deleting them entirely from the feature space might – while preventing over-reliance on them – hurt prediction quality considerably. However, we find a way to blur the signals of the LFs instead of removing them.

In this work we propose KnowMAN (Knowledge-based Weakly Supervised Multinomial Adversarial Networks), a method for controllable *soft deletion* of LF signals, allowing a trade-off between reliance and generalization. Inspired by adversarial learning for domain adaptation (Chen and Cardie, 2018a; Ganin and Lempitsky, 2015), we consider LFs as domains and aim to learn a *LF-invariant* feature extractor in our model. KnowMAN is composed of three modules: a feature extractor, a classifier, and a discriminator. Specifically, KnowMAN employs a classifier that learns the actual task and an adversarial opponent, the LF- discriminator, that learns to distinguish between the different LFs. Upstream of both is the shared feature extractor to which the gradient of the classifier and the reversed gradient of the discriminator are propagated. In our experiments, the feature extractor for encoding the input is a multi-layer perceptron on top of either a bag-of-words vector or a transformer architecture, but KnowMAN is in principle usable with any differentiable feature extractor.

KnowMAN consistently outperforms our baselines by 2 to 30% depending on the dataset. By setting a hyperparameter λ that controls the influence of the adversarial part we can control the degree of

Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pages 9549–9557
November 7–11, 2021.

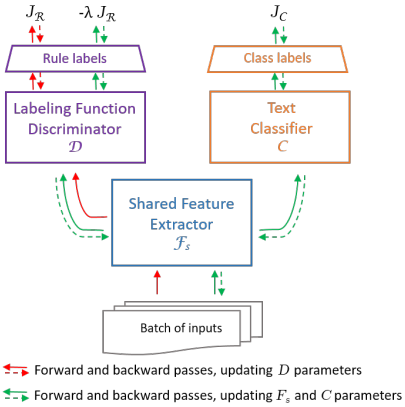


Figure 1: KnowMAN architecture. The figure depicts one iteration over a batch of inputs. The parameters of $\mathcal{C}$ and $\mathcal{F}_s$ are updated together, following the green arrows. The LF discriminator $\mathcal{D}$ is updated following the red arrows. Solid lines indicate forward, dashed lines the backward pass.

discarding the information of LF-specific signals. The optimal λ value depends on the dataset and its properties.

The contributions of this work are i) proposing an adversarial architecture for controlling the influence of signals associated with specific LFs, ii) consistent improvements over weakly supervised baselines, iii) release of our code [1]. To our knowledge, we are the first that apply adversarial learning to overcome the noisiness of labels in weak supervision.

2 Method

Our approach is composed of three interacting modules i) the shared feature extractor $\mathcal{F}_s$, ii) the classifier $\mathcal{C}$ and iii) the LF discriminator $\mathcal{D}$. The loss function of $\mathcal{C}$ rewards the classifier $\mathcal{C}$ for predicting the correct label for the instance, and the gradient is used for optimizing the shared feature extractor and classifier modules towards that goal. At the same time, the loss function for the LF-discriminator $\mathcal{D}$ rewards predicting which LF was responsible for labeling an instance. However, in adversarial optimization, KnowMAN backpropagates the *reversed* gradient for the LF-discriminator, hence the information indicative for distinguishing between specific LFs is weakened throughout the network. The hyperparameter λ is used to control the level of weakening the signals - the higher we choose the value the more influence is assigned to the discriminator information that goes into $\mathcal{D}$. The result of the interplay between classifier and LF-discriminator is a shared feature representation that is good at predicting the labels while reducing the influence of LF-specific signals, encouraging the shared feature extractor to take other information (correlated with all LFs for a class) into account.

In Figure 1, the arrows illustrate the training flow of the three modules. Due to the adversarial nature of the LF discriminator $\mathcal{D}$, it has to be trained with a separate optimizer (red arrows), while the rest of the network is updated with the main optimizer (green arrows). When $\mathcal{D}$ is trained the parameters of $\mathcal{C}$ and $\mathcal{F}_s$ are frozen and vice versa.

To calculate the losses we utilize canonical negative log-likelihood loss (NLL) and use it for both, the classifier and the LF discriminator. The classification NLL can be formalized as:

$$\mathcal{L}_C(\hat{y}_i, y_i) = -\log P(\hat{y}_i = y_i) \quad (1)$$

where y_i is the (weakly supervised) annotated label and $\hat{y}_i$ is the prediction of the classifier module $\mathcal{C}$, for a training sample i. Analogously, we can define the NLL for the LF discriminator:

$$\mathcal{L}_{\mathcal{D}}(\hat{lf}_i, lf_i) = -\log P(\hat{lf}_i = lf_i) \quad (2)$$

where lf_i is the actual LF used for annotating sample i and $\hat{lf}_i$ is the predicted LF by the discriminator $\mathcal{D}$. Accordingly, we minimize two different objectives within KnowMAN:

$$J_{\mathcal{C}} = \sum_{i=1}^{N} \mathcal{L}_{\mathcal{C}}(\mathcal{C}(\mathcal{F}_s(x_i); y_i)) \quad (3)$$

$$J_{\mathcal{D}} = \sum_{i=1}^{N} \mathcal{L}_{\mathcal{D}}(\mathcal{D}(\mathcal{F}_s(x_i); lf_i)) \quad (4)$$

Here the shared feature extractor has two different objectives: i) help $\mathcal{C}$ to achieve better classification performance and ii) make the feature distribution invariant to the signals from the LFs. This is captured by the shared objective:

$$J_{\mathcal{F}_s} = J_{\mathcal{C}} + \lambda \cdot (-J_{\mathcal{D}}) \quad (5)$$

where λ is the parameter that controls the adversarial influence i.e. the degree of LF signal blur. $-J_{\mathcal{D}}$ is the reversed loss of the LF discriminator $\mathcal{D}$ that represents $\mathcal{C}s$ adversarial opponent. In general, the

[1] https://github.com/LuisaMaerz/KnowMAN

exact implementation or architecture of the individual modules is interchangeable and can be set up as required. This makes KnowMAN a universally applicable and easily customizable architecture.

3 Experiments

3.1 Data

For our experiments we use three standard datasets for weak supervision.

Spam. Based on the YouTube comments dataset (Alberto et al., 2015) there is a smaller Spam dataset from Snorkel (Ratner et al., 2020) where the task is to classify if a text is relevant to a certain YouTube video or contains spam. This dataset is very small and does consist of a train and a test set only. For the 10 LFs keywords and regular expressions are used.

Spouse. This dataset for extracting the *spouse* relation has also been created by Snorkel, it is based on the Signal Media One-Million News Articles Dataset (Corney et al., 2016). The 9 LFs use information from a knowledge base, keywords and patterns. One peculiarity of this dataset is that over 90% of the instances do not hold a spouse relation.

IMDb. The IMDb dataset contains movie reviews that should be classified in terms of their sentiment (binary, positive or negative sentiment). The LFs used for this dataset are occurrences of positive and negative keywords from (Hu and Liu, 2004). A particular characteristic of this data set is the large amount of 6800 LFs, which constitutes a particular challenge to the Snorkel denoising framework. As a result Snorkel fails to calculate its generative model, since its memory consumption exceeds the available limit of 32GB RAM.

3.2 Experimental setup

For the experiments we use two different methods for encoding the input: i) TF-IDF encoding and ii) a DistilBERT transformer. For TF-IDF encoding, we vectorize[2] the input sentences and feed them to a simple MLP. In the transformer setting, the sequences of words are encoded using a pre-trained DistilBERT. Similar to BERT (Devlin et al., 2019), DistilBERT is a masked transformer language model, which is a smaller, lighter, and faster version leveraging knowledge distillation while retaining 97% of BERT's language understanding capabilities (Sanh et al., 2019).

Our encoder takes the representation of the CLS token from a frozen DistilBERT and learns a non-linear transformation with a drop-out layer to avoid overfitting (Srivastava et al., 2014):

$$h_i = DistilBERT(Sentence_i)_{[CLS]}$$

$$F_{si} = Dropout(ReLU(f(h_i)))$$

where $DistilBERT(.)_{[CLS]}$ generates the hidden state of the BERT's classifier token (CLS) and the function f represents a linear transformation for the i^{th} sentence.

The classifier and discriminator networks following the feature extractor are in line with the implementation of Chen and Cardie (2018a) for domain-adversarial learning. Both are simple sequential models with dropout, batch normalization, $ReLU$ activation and softmax as the last layer. Please see our code for implementation details. In the TF-IDF setup we use Adam (Kingma and Ba, 2014) for both optimizers. When using transformer encoding the $\mathcal{D}$ optimizer again is Adam and the $\mathcal{C}$ optimizer is AdamW (Loshchilov and Hutter, 2018), as this yielded more stable results.

Baselines For each input encoding we implemented several baselines. Weakly supervised TF-IDF (*WS TF-IDF*) and Weakly supervised DistilBERT (*WS DistilBERT*). Both calculate the labels for each instance in the train set based on their matching LFs. *WS TF-IDF* directly applies a logistic regression classifier to the input and the calculated labels. *WS DistilBERT* directly uses the DistilBERT uncased model for English (Sanh et al., 2019) as a prediction model. The second baseline (*Feature TF-IDF*, *Feature DistilBERT*) uses feature extractor and classifier layers of KnowMAN without taking the information of $\mathcal{D}$ into account (this is equal to setting λ to zero). We also fine-tuned the pure language model (*Fine-tuned DistilBERT*) without further transformations and without integrating the KnowMAN architecture.

We also compare with training *TF-IDF* and *DistilBERT* models on labels denoised by Snorke (*Snorkel TF-IDF*, *Snorkel DistilBERT*). However, Snorkel denoising failed for the IMDb data set due to the large amount of LFs.

KnowMAN We refer to the KnowMAN architecture as *TF-IDF KnowMAN* and *DistilBERT KnowMAN*. Depending on the dataset we choose different λ values. We also implemented two ways

[2] https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html

	Spam	Spouse			IMDb
	Acc	P	R	F1	Acc
WS TF-IDF	0.87	0.12	**0.83**	0.20*	0.65*
Feature TF-IDF	0.91	0.12	0.76	0.21*	0.75*
Snorkel TF-IDF	0.81	0.18	0.63	0.28*	0.50*
KnowMAN TF-IDF	**0.94**	**0.16**	0.72	**0.35**	**0.77**
Fine-tuned DistilBERT	**0.92**	0.14	0.78	0.24	0.70
WS DistilBERT	0.87	0.09	**0.90**	0.17*	0.67*
Feature DistilBERT	0.86	0.18	0.80	0.29*	0.74
Snorkel DistilBERT	0.88	0.13	0.70	0.23*	0.49*
KnowMAN DistilBERT	0.90	**0.27**	0.67	**0.39**	**0.76**

Table 1: Results on the test sets. The * indicates that KnowMAN performs significantly better than the marked model. For the Spouse data set we do report significance for the F1 scores only.

of evaluation and best model saving during training: i) evaluate after each batch and save the best model, ii) evaluate after a certain number of steps in between the batches and save the best model.

Hyperparameters We perform hyperparameter tuning using Bayesian optimization (Snoek et al., 2012) for the IMDb and Spouse datasets. For Spam, hyperparameters are not optimized, as no validation set is available. Sampling history and resulting hyperparameters are reported in the Appendix, Figures 2, 3 as well as hyperparameters chosen for the Spam data set.

Evaluation For the evaluation of the IMDb and the Spam datasets we use accuracy, for the Spouse dataset we use the macro F1 score of the positive class. To check statistical significance we use randomized testing (Yeh, 2000). Results are considered significant if $\rho < 0.05$.

3.3 Results

The results of the experiments are shown in Table 1. For the TF-IDF setup *KnowMAN TF-IDF* outperforms the baselines across all datasets. We find the optimal λ values as follows: Spam/Spouse/IMDb = 2/5/4.9. Using the additional feature extractor layer (*Feature TF-IDF*) is beneficial compared to direct logistic regression for all datasets. *Snorkel TF-IDF* can outperform the other two baselines for the Spouse dataset only.

Fine tuning of DistilBERT can not outperform our best KnowMAN. However, for the Spam dataset *Fine-tuned DistilBERT* gives better results than *KnowMAN DistilBERT* but still is worse than *KnowMAN TF-IDF*. Using *WS DistilBERT* gives the same results for the Spam dataset and slightly better results for IMDb, when compared to *WS TF-IDF*, for Spouse the performance decreases. *Snorkel DistilBERT* can outperform the other two baselines for the Spam dataset only. The low performance of Snorkel on IMDb (for both DistilBERT and TF-IDF) might be explained by the very large amount of LF for this dataset. The *KnowMAN DistilBERT* results across datasets are in line with the TF-IDF setup - KnowMAN can outperform all baselines for the Spouse and IMDb dataset. We observe that $\lambda = 5$ for Spouse and $\lambda = 1$ for IMDb is most beneficial when using DistilBERT. For the Spam dataset we observe that KnowMAN (with $\lambda = 2$) outperforms all the baselines, except for the fine-tuned DistilBERT model.

Discussion The performance drop we observe with DistilBERT for KnowMAN compared to the tf-idf setup of the IMDb dataset could be explained by implementation details. Due to memory issues we have to truncate the input when using DistilBERT. Since the movie reviews from IMDb are rather long this could harm performance. Since the Spam dataset is very small a single wrongly classified instance can have great impact on the results. This could explain why *KnowMAN TF-IDF* outperforms *KnowMAN DistilBERT* here as well. In general we could not perform hyperparameter optimization for the DistilBERT experiments due to memory issues. Therefore the results for that experiments might not have reached their optimum. However, the results show the value of using KnowMAN though. Overall our results confirm the assumption that KnowMAN enables a focus shift of the shared feature extractor from the signals of the LFs towards signals of other valuable information. KnowMAN consistently improves over the other experiments significantly - except

for the Spam dataset. We assume that the dataset size is too small to see significant changes in the results. Compared to the implementation of Chen and Cardie (2018a) we could not use the specialized domain feature extractor for our datasets in the experiments. This is due to the fact that our test sets do not contain information about LF matches. However, we will address this issue by integrating a mixture of experts module for the specialized feature extractor as recommended by Chen et al. (2019).

4 Related Work

Adversarial neural networks have been used to reduce the divergence between distributions, such as Goodfellow et al. (2014), Chen et al. (2018) and Ganin and Lempitsky (2015). The latter proposed an architecture for gradient reversal and a shared feature extractor. Unlike us, they focused on a binary domain discriminator. Similarly, (Chen and Cardie, 2018a) use an adversarial approach in a multinomial scenario for domain adaptation.

Some works on adversarial learning in the context of weak supervision focus on different aspects and only share similarity in name with our approach: Wu et al. (2017) use *virtual adversarial training* (Miyato et al., 2017) for perturbing input representations, which can be viewed as a general regularization technique not specific to weakly supervised learning. Qin et al. (2018); Zeng et al. (2018) use generative adversarial mechanisms for selecting *negative* training instances that are difficult to discriminate from heuristically annotated ones for a classifier.

Several approaches have focused on denoising the labels for weakly supervised learning (Takamatsu et al., 2012; Manning et al., 2014; Lin et al., 2016). Snorkel (Ratner et al., 2020) is one of the most general approaches in this line of work. However, Snorkel only models biases and correlations of LFs, and does not consider problems of weak supervision that may stem from biases in the features and learned representations.

A recent approach that focuses on denoising weakly supervised data is (Sedova et al., 2021). Knodle is a framework for comparison of different methods that improve weakly supervised learning. We use some of their datasets for our approach but denoise the signals of the LFs during training.

5 Conclusion

We propose KnowMAN - an adversarial neural network for training models with noisy weakly supervised data. By integrating a shared feature extractor that learns labeling function invariant features, KnowMAN can improve results on weakly supervised data drastically across all experiments and datasets in our setup. The experiments also show that the adverse effect of labeling function-specific signals is highly dependent on the datasets and their properties. Therefore, it is crucial to fine-tune the λ parameter on a validation set to find the optimal degree of blurring the labeling function signals. Since the modules in the KnowMAN architecture are easily exchangeable, KnowMAN can be applied to any architecture and dataset labeled with heuristic labeling functions.

Acknowledgements

This research was funded by the WWTF through theproject "Knowledge-infused Deep Learning for Nat-ural Language Processing" (WWTF Vienna ResearchGroup VRG19-008), by the Deutsche Forschungs-gemeinschaft (DFG, German Research Foundation) -RO 5127/2-1.

References

Tulio Alberto, Johannes Lochter, and Tiago Almeida. 2015. Tubespam: Comment spam filtering on youtube. pages 138–143.

Christoph Alt, Marc Hübner, and Leonhard Hennig. 2019. Improving relation extraction by pre-trained language representations. In *Automated Knowledge Base Construction (AKBC)*.

Xilun Chen, Ahmed Hassan Awadallah, Hany Hassan, Wei Wang, and Claire Cardie. 2019. Multi-source cross-lingual model transfer: Learning what to share. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3098–3112, Florence, Italy. Association for Computational Linguistics.

Xilun Chen and Claire Cardie. 2018a. Multinomial adversarial networks for multi-domain text classification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1226–1240, New Orleans, Louisiana. Association for Computational Linguistics.

Xilun Chen and Claire Cardie. 2018b. Multinomial adversarial networks for multi-domain text classification. In *Proceedings of the 2018 Conference of the*

North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers), pages 1226–1240, New Orleans, Louisiana. Association for Computational Linguistics.

Xilun Chen, Yu Sun, Ben Athiwaratkun, Claire Cardie, and Kilian Weinberger. 2018. Adversarial deep averaging networks for cross-lingual sentiment classification. volume 6, pages 557–570.

D. Corney, M. Albakour, Miguel Martinez-Alvarez, and Samir Moussa. 2016. What do a million news articles look like? In *NewsIR@ECIR*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Yaroslav Ganin and Victor Lempitsky. 2015. Unsupervised domain adaptation by backpropagation. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, page 1180–1189. JMLR.org.

Alec Go, Richa Bhayani, and Lei Huang. 2009. Twitter sentiment classification using distant supervision. *CS224N project report, Stanford*, 1(12):2009.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. In *NIPS*.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '04, page 168–177, New York, NY, USA. Association for Computing Machinery.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *International Conference on Learning Representations*.

Yankai Lin, Shiqi Shen, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. 2016. Neural relation extraction with selective attention over instances. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2124–2133.

Ilya Loshchilov and Frank Hutter. 2018. Fixing weight decay regularization in adam.

Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*, pages 55–60.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 1003–1011.

Takeru Miyato, Andrew M. Dai, and Ian Goodfellow. 2017. Adversarial training methods for semi-supervised text classification.

Pengda Qin, Weiran Xu, and William Yang Wang. 2018. DSGAN: Generative adversarial training for distant supervision relation extraction. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 496–505, Melbourne, Australia. Association for Computational Linguistics.

Alexander Ratner, Stephen H. Bach, Henry R. Ehrenberg, Jason A. Fries, Sen Wu, and Christopher Ré. 2020. Snorkel: rapid training data creation with weak supervision. *VLDB J.*, 29(2-3):709–730.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of BERT: smaller, faster, cheaper and lighter. *CoRR*, abs/1910.01108.

Anastasiya Sedova, Andreas Stephan, Marina Speranskaya, and Benjamin Roth. 2021. Knodle: Modular weakly supervised learning with pytorch. *CoRR*, abs/2104.11557.

Jasper Snoek, Hugo Larochelle, and Ryan Adams. 2012. Practical bayesian optimization of machine learning algorithms. In *Proc. NIPS*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958.

Shingo Takamatsu, Issei Sato, and Hiroshi Nakagawa. 2012. Reducing wrong labels in distant supervision for relation extraction. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 721–729.

Yi Wu, David Bamman, and Stuart Russell. 2017. Adversarial training for relation extraction. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1778–1783.

Alexander Yeh. 2000. More accurate tests for the statistical significance of result differences. In *COLING 2000 Volume 2: The 18th International Conference on Computational Linguistics*.

Daojian Zeng, Yuan Dai, Feng Li, R Simon Sherratt, and Jin Wang. 2018. Adversarial learning for distant supervised relation extraction. *Computers, Materials & Continua*, 55(1):121–136.

A Appendix

A.1 Dataset statistics

The datasets used for the KnowMAN experiments have different properties. Especially the numer of labeling functions and the dataset sizes varies a lot.

dataset	classes	train/test samples	lfs
Spam	2	1586/250	10
Spouse	2	22254/2701	9
IMDb	2	40000/5000	6786

Table 2: Dataset statistics for KnowMAN experiments. Lfs are labeling functions.

A.2 Hyperparameter optimization

We perform hyperparameter tuning using Bayesian optimization (Snoek et al., 2012). Bayesian Optimization is an approach that uses the Bayes Theorem to direct the search in order to find the minimum or maximum of a black-box objective function. In comparison with random search and grid search, it tends to obtain better hyperparameters in fewer steps by making a proper balance between exploration and exploitation steps. Our hyperparameter space includes batch size, dropout, number of iterations over $\mathcal{D}$, the shared hidden size of the models, learning rate for $\mathcal{D}$ and $\mathcal{F}_s, \mathcal{C}$ and the number of layers of $\mathcal{C}, \mathcal{D}$ and $\mathcal{F}_s$. We implemented two ways of evaluation and best model saving during training: i) evaluate after each batch and save the best model, ii) evaluate after a certain number of steps in between the batches and save the best model. We also optimized the number of steps if logging in between a batch.

We evaluated the models for IMDb and Spouse on the respective validation set. For the Spam dataset, there is no development set available and we used the following hyperparameters for *KnowMAN TF-IDF* following the parameters used in Chen and Cardie (2018b): **Batch size:** 32, **dropout:** 0.4, **n critic:** 5, **lambda:** 2.0, **shared hidden size:** 700, **learning rate C & F:** 0.0001, **learning rate D:** 0.0001 , **number of F layers:** 1, **number of C layers:** 1, **number of D layers:** 1.

A.3 Experimental details

We ran our experiments on a DGX-1 server with one V100 GPU per experiment. The runtime of one model depends on the dataset: 0.25 hours for the Spam dataset, 0.25 hours for the Spouse dataset, and 8 hours for the IMDb dataset.

Please find our implementation at https://github.com/LuisaMaerz/KnowMAN.

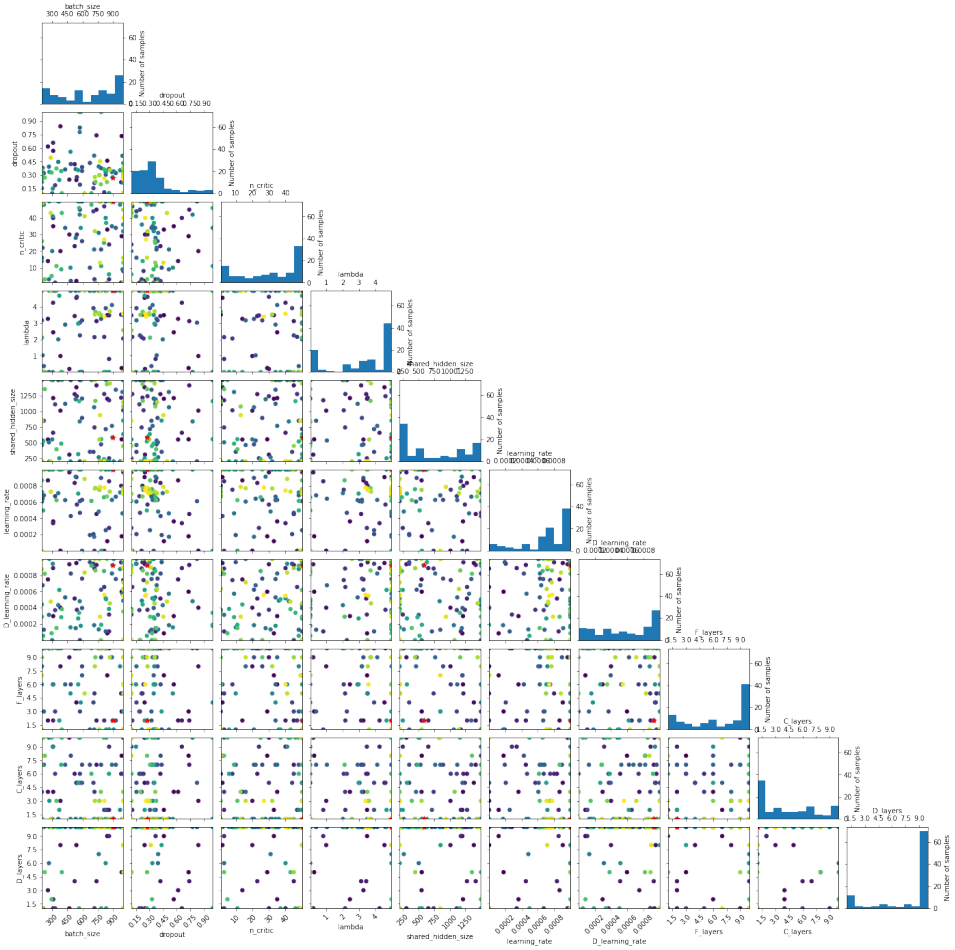


Figure 2: Sampled hyperparameters for KnowMAN TF-IDF on IMDb. Optimal hyperparameters are indicated in red.
Batch size: 895, **dropout:** 0.275, **n critic:** 50, **lambda:** 4.9, **shared hidden size:** 585, **learning rate C & F:** 0.0001, **learning rate D:** 0.0001, **number of F layers:** 1 , **number of C layers:** 1, **number of D layers:** 10.
Histograms on the diagonal show how, for each hyperparameter, how many samples have been drawn during optimization.

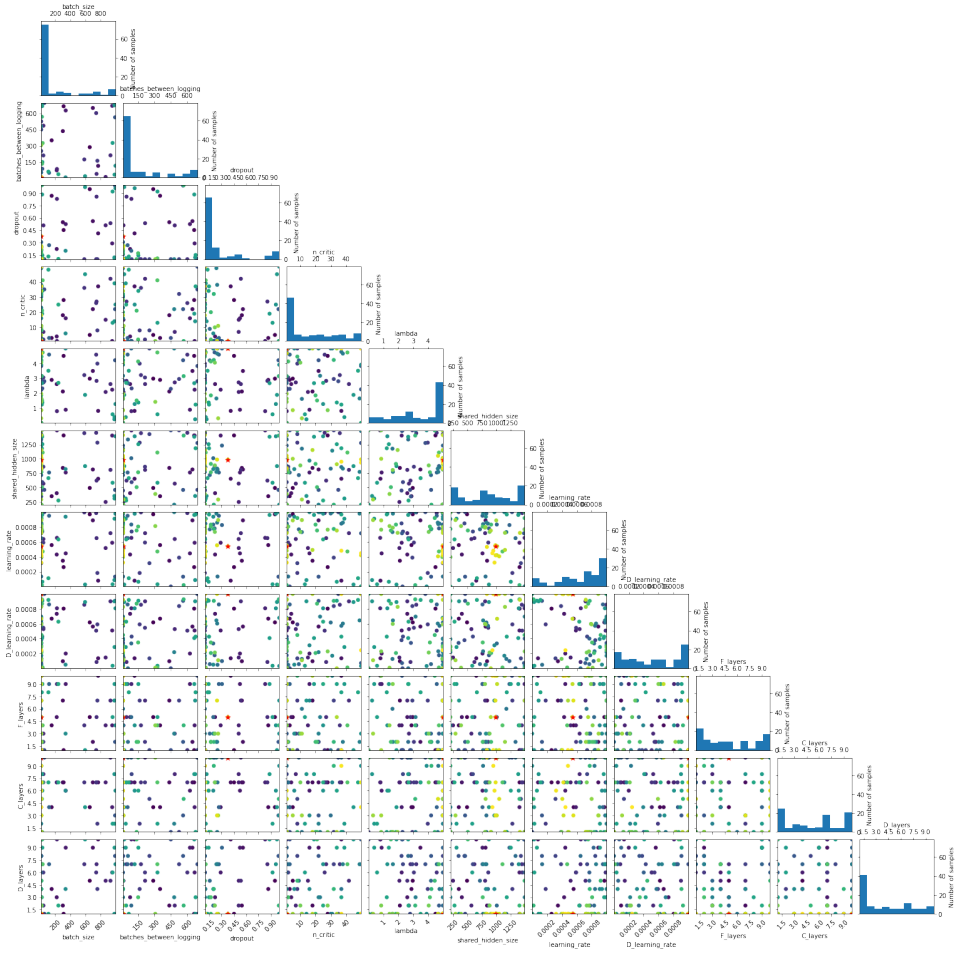
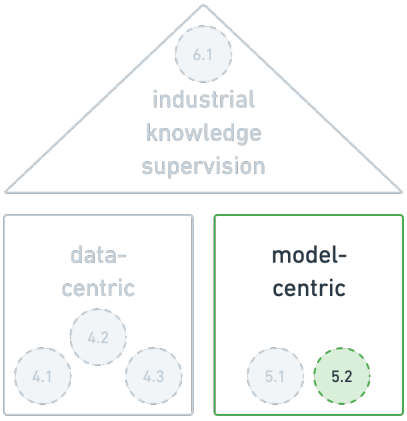


Figure 3: Sampled hyperparameters for KnowMAN DistilBERT on Spouse. Optimal hyperparameters are indicated in red.
Batch size: 16, **dropout:** 0.379, **n critic:** 1, **lambda:** 5.0, **shared hidden size:** 988, **learning rate C & F:** 0.0005, **learning rate D:** 0.001 , **number of F layers:** 5, **number of C layers:** 10, **number of D layers:** 1.
Histograms on the diagonal show how, for each hyperparameter, how many samples have been drawn during optimization.

5.2. XPASC: Measuring Generalization in Weak Supervision



Contributing Article

Submitted to *Journal of Natural Language Engineering* on 06 May 2022, accepted (with minor revisions) on 03 September 2022, re-submitted on 22 November 2022.

März, L., Asgari, E., Braune, F., Zimmermann, F., & Roth, B. (2022). XPASC: Measuring Generalization in Weak Supervision by Explainability and Association. *ArXive preprint:* `https://arxiv.org/abs/2206.01444`

Copyright Information

Author Contributions

The method and conception of the work were developed in discussions between Luisa März and Benjamin Roth. Programming was done by Luisa März and includes data compilation, data pre-processing, modules for processing data and using models from different source algorithms, modules for computing association and explainability, implementation of the XPASC measure, as well as scripts for evaluation and analysis. Ehsan Asgari, Fabienne Braune, and Franziska Zimmermann contributed with constant feedback and discussion on the method and the implementation. The writing of the manuscript was done by Luisa März, with continuous revision from the other authors. Besides the authors, Andreas Stephan and Pedro Henrique Luz de Araujo also contributed with feedback on the manuscript. Luisa März and Benjamin Roth finally revised and proofread the manuscript together.

Supplementary Material

`https://github.com/LuisaMaerz/XPASC`

Journal (2022), 1–00
doi:10.1017/xxxxx

ARTICLE

XPASC: Measuring Generalization in Weak Supervision by Explainability and Association

Luisa März[◇,*,⋆], Ehsaneddin Asgari[*], Fabienne Braune[*], Franziska Zimmermann[○], and Benjamin Roth[◇,†]

[◇] Research Group Data Mining and Machine Learning,
Faculty of Computer Science, University of Vienna,
Vienna, Austria

[†] Faculty of Philological and Cultural Studies,
University of Vienna,
Vienna, Austria

[*] AI Innovation & Pre-Development,
Data:Lab, Volkswagen AG,
Munich, Germany
luisa.k.maerz@gmail.com

[⋆] UniVie Docotoral School Computer Science,
Vienna, Austria

[○] CAPE Analytics,
Mountain View, California, United States

(Received 06 May 2022; revised 22 November 2022; accepted xx xxx xxx)

Abstract

Weak supervision is leveraged in a wide range of domains and tasks due to its ability to create massive amounts of labeled data, requiring only little manual effort. Standard approaches use labeling functions to specify signals that are relevant for the labeling. It has been conjectured that weakly supervised models over-rely on those signals and as a result suffer from overfitting. To verify this assumption, we introduce a novel method, XPASC (eXPlainability-Association SCore), for measuring the generalization of a model trained with a weakly supervised dataset. Considering the occurrences of features, classes and labeling functions in a dataset, XPASC takes into account the relevance of each feature for the predictions (explainability) of the model as well as the connection of the feature with the class and the labeling function (association), respectively. The explainability is measured using occlusion in this work. The association in XPASC can be measured in two variants: XPASC-CHI SQUARE measures associations relative to their statistical significance, while XPASC-PPMI measures association strength more generally.

We use XPASC to analyze KNOWMAN, an adversarial architecture intended to control the degree of generalization from the labeling functions and thus to mitigate the problem of overfitting. On one hand, we show that KNOWMAN is able to control the degree of generalization through a hyperparameter. On the other hand, results and qualitative analysis show that generalization and performance do not relate one-to-one, and that the highest degree of generalization does not necessarily imply the best performance. Therefore methods that allow for controlling the amount of generalization can achieve the right degree of benign overfitting. Our contributions in this study are i) the XPASC score to measure generalization in weakly-supervised models, ii) evaluation of XPASC across datasets and models and iii) the release of the XPASC implementation.

1. Introduction

Many machine learning architectures still require large amounts of labeled training data, resulting in static data sets with limited usability for changing data distributions or task definitions. Manual annotation is both expensive and time-consuming and thus not always practically feasible or convenient. One way to circumvent this problem is to use weak supervision. Weak supervision methods utilize different knowledge sources such as knowledge bases, heuristics, or taxonomies to annotate large amounts of data automatically. The knowledge of the external sources is encoded in *labeling functions*, programmatically specified heuristics (e.g., keywords, patterns, database lookups) that trigger the automatic annotation of a specific output. A labeling function can also be thought of as a decision rule that is defined based on prior (expert) knowledge. If this rule is matched, the appropriate output is assigned to the instance. Due to the fact that labeling functions only consider a narrow context for triggering an annotation, it is likely that some weak labels are noisy and/or imprecise. Moreover, large sets of weakly annotated instances all follow similar patterns (captured by the same labeling function), and it has been conjectured that weakly supervised models rely too heavily on the labeling functions and therefore suffer from overfitting (März et al. 2021; Dehghani et al. 2017).

Approaches to tackle the problem of noisy weak labels can be categorized into two main groups: Those who try to filter out the noisy labels for training (Ren et al. 2020; Sukhbaatar et al. 2014; Dehghani et al. 2017) and those who try to estimate the accuracy of the labeling functions or the weak sources (Fu et al. 2020; Ratner et al. 2020). See Zhang et al. (2022a) for a detailed survey on weak supervision approaches and labeling function modeling. None of them addresses the problem of overfitting to (or, inversely, generalization from) labeling functions. As an alternative, we developed KNOWMAN (März et al. 2021), an adversarial architecture with the objective to shift the focus for the learned representation of a model away from the labeling functions towards a more general representation. This is achieved through a hyperparameter that controls the influence of the labeling functions on the feature representation. KNOWMAN has explicitly been designed to overcome the problem of overfitting to the noisy labeling function signals. However, while training in the KNOWMAN-settings increases the prediction quality for the studied weakly supervised neural networks, it could not be *directly* evaluated whether the KNOWMAN actually increases generalization from the labeling functions.

In this study, we present **XPASC (eXPlainability-Association SCore)** to observe generalization from noisy signals in weakly supervised models more closely. The intuition behind the score is that models suffering from overfitting to labeling functions have a low ability to generalize, and will heavily rely on features associated with labeling functions for prediction. Generalization in the scope of this work means the capability of a model to abstract from the labeling function signals and to learn representations based on various signals and parts of the input. A higher generalization should ultimately lead to the representation being more robust against misleading labeling functions and being able to represent the input with as many aspects as possible. Accordingly, a greater generalization from the weak source indicates a smaller degree of overfitting. In this work we consider the strongly information-carrying surface forms, i.e. the individual tokens, of an instance as features. By using XPASC the generalization ability of a model, given a data set, can be measured. The score combines the relevance of each feature for the prediction of a weakly supervised model (explainability) with the connection of the feature with the class and the labeling function (association).

To measure prediction relevance of the single features we leverage a method from XAI (eXplainable AI), namely occlusion. In general, XAI methods aim to give insights into the output of machine learning models to make internal processes more transparent to users. Taking into account the explainability method is central to the analysis with XPASC, and also represents an unconventional use case of XAI.

To compute association in XPASC we propose two different methods: XPASC-CHI SQUARE which relies on statistical association strength, and XPASC-PPMI which measures the more general information-theoretic association strength.

Apart from introducing the formal details of XPASC, we study KNOWMAN and WEASEL as well as two traditional weak supervision approaches, MAJORITY VOTE and SNORKEL-DP (Data Programming), with respect to their generalization, as measured by XPASC. Our findings show that the KNOWMAN architecture is able to control the degree of generalization in direct relation to its hyperparameter λ. In fact, KNOWMAN can get the model to focus more on words that are associated with the class (generalize more), and even ignore features highly associated with single, misleading labeling functions.

We have observed that the generalization of a model and its performance are not one-to-one related, and that at a certain point there can be too much generalization from the weak signals. Our observations show that the two different association computations, PPMI and CHI SQUARE, behave analogously in the overall picture. However, we find that the values of the association based on PPMI are distributed across a larger space. This means that PPMI also takes into account the "long tail" of data sets and presents the reality in the data more straightforwardly. The values of the CHI SQUARE-based association, on the other hand, have a denser distribution. So CHI SQUARE also appears to be more resilient to outliers in the data.

Our main contributions in this paper are:

- the proposal and detailed introduction of XPASC to measure generalization from weak signals
- the evaluation of XPASC across models and data sets
- the confirmation of the hypothesis and functionality of KNOWMAN
- the release of the XPASC implementation[a].

The remainder of the paper is structured as follows: After investigating related work in the fields of weak supervision and overfitting metrics we describe the method the XPASC formally. Section §4 gives an overview over models and data sets used in this work. The analysis is divided in quantitative and linguistic results and followed by the conclusion.

2. Related Work

We consider the concepts of overfitting and generalization to be related in that overfitting can be a result of low generalization. In the weak supervision context, this means that a model that abstracts little from the labeling functions, i.e. has low generalization, is more likely to overfit to these misleading signals. In the following we present approaches that focus either on overfitting or on generalization. Some are tailored to weak supervision, while others deal with overfitting and generalization in natural language processing in general.

Overfitting of machine learning models has been repeatedly identified as a problem. In machine learning generally, overfitting means that a model has adapted too much to the training data and can therefore no longer perform well on newly seen data. Model performance on unseen data or held-out test sets is therefore typically used as an indicator for overfitting. If the result on unseen data is much worse than on training data it is likely that the model overfitted to the training data. Salman and Liu (2019) analyze the training dynamics and the application of models to unseen data to observe overfitting. Other works measure overfitting by the total number of parameters, where a low number of parameters indicates less overfitting than a high number of parameters. (Li et al. 2019) measure overfitting through the number of parameters of a model and its accuracy on the test set.

[a] `https://github.com/LuisaMaerz/XPASC`

Roelofs et al. (2019) analyze overfitting caused by test set reuse on a large set of Kaggle competitions. The assumption is that if many models are centered towards one test set, overfitting of the models to that test set is likely. However, with their experiments they show that there is no significant overfitting due to test set reuse in Kaggle. Roelofs et al. (2019) provide a simple measurement for the adaptivity gap between the losses on train and test set. Here they use the fact that Kaggle provides two types of test sets with which this gap can be approximated very well. Unlike us, they cover overfitting to test set reuse rather than overfitting to labeling functions.

Due to the nature of weak supervision, models may overfit to systematic errors and biases introduced by the automatic labeling process. Yu et al. (2021) fine tune a pre-trained language model with weak supervision. This is challenging, because large language models have a higher risk to overfit due to their large amount of parameters anyways. Errors are more propagated due to overfitting, which degrades performance and prevents the models from learning properly. Yu et al. (2021) tackle this issue by contrastive self-training, what can be considered as denoising in the first place and reduces error propagation and overfitting to the noise. In their experiments, they use RoBERTa (Liu et al. 2019) and fine-tune a simple classification head. Like us, they use MAJORITY VOTE (*exact match* in their work) and SNORKEL-DP for weak labeling. However, they do not specifically address the impact of overfitting to labeling functions. As in our previous work with KNOWMAN (März et al. 2021), their model aims at learning better representations from weakly labeled data. Unlike us, they use a contrastive approach that pulls labels with similar weak supervision signals closer together and pushes others further away in the feature space, rather than an adversarial network as in KNOWMAN. In recent work Zhang et al. (2022b) propose the source-aware Influence Function to understand programmatic weak supervision. By observing changes in the loss of a model while utilizing the source-aware influence function, they gain insights in the influence of single data points, labeling functions or weak sources on model performance. Like us, they aim to identify important parts of the input to make the model output more explainable. Similar to KNOWMAN, they try to reduce the impact of misleading labeling functions on model training. Although they provide some insight into what influences the training through the source-aware influence function, they do not use this information to compare different models, which is different from our work.

Generalization refers to a model's ability to perform well on unseen data, i.e., a model generalizes well if it overfits only slightly. For example Ratner et al. (2019) consider generalization of weak supervision sources observable through the estimation error of their trustworthiness. They claim that the generalization error scales with the number of unlabeled data points and try to minimize the loss for predicting weak labels without loss of generality. By connecting generalization to the estimation error, generalization is not only observable, but also controllable. Like our metric, this can be viewed as a formal measurement of generalization. Unlike our metric, their measure is tightly coupled with their specific weak supervision approach and not generally applicable as a universal tool to compare generalization across models and data sets.

Many weak supervision approaches try to overcome a lack of generalization by denoising the weak sources (Ren et al. 2020; Hsieh et al. 2022). In contrast to these approaches, we address the issue of generalizing from weak supervision sources, instead of denoising them. Fu et al. (2020) provide a weak supervision framework to model and label data by leveraging different weak supervision sources. In addition, they provide a bound for generalization. To do so, they measure the performance gap between the end model parametrization using outputs of the label model and the optimal end model parametrization over the true distribution of labels. More efforts can be mentioned in studying generalization in general, e.g., measuring number of required "strong" labels (Robinson et al. 2020), studying generalization in algorithmic datasets (Power et al. 2022), generalization in generative models by measuring the uniqueness of generated sample (Mauri et al. 2022).

Although there is work on both overfitting and generalization, and both are considered to be important issues in weak supervision, to the best of our knowledge XPASC is the first universal

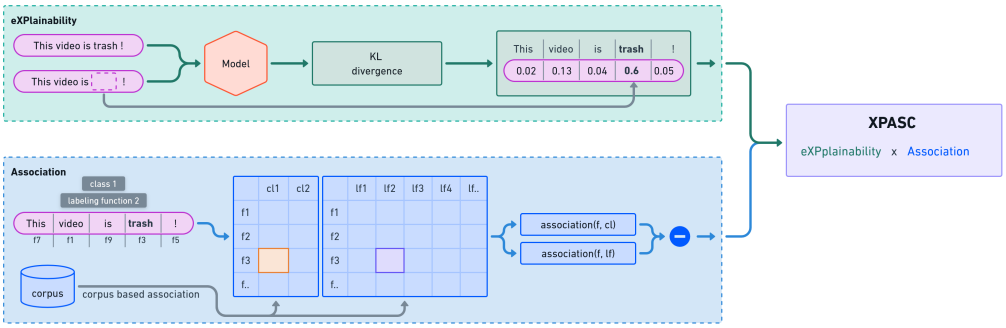


Figure 1. Modules of XPASC: product of *eXPlainability* and *Association*. The *eXPlainability* of a feature of an instance is calculated by the KL divergence of the class predictions for the entire instance and the instance without the respective feature. For each instance the explainability for all features is computed and the most important feature can be obtained. The *Association* for an instance given a feature is calculated as the difference of the association of the feature with the class and with the labeling function, which in practice is a matrix lookup. The matrices are computed in advance and are based on coocurrences of features, classes and labeling functions in the data corpus.

score measuring overfitting to and generalization from labeling functions in weakly supervised models. Moreover, since our approach is based on explainability methods, it makes transparent which features are mainly responsible for overfitting or generalization.

3. The Explainability-Association Score

The goal of XPASC is to measure generalization from weak signals for weakly supervised models. The intuition behind XPASC is that input parts or features that are most important for the class prediction of these models are highly associated with the heuristics used for annotating the training data. This is due to the fact that the model relies too much on the labeling functions and therefore tends to ignore other valuable signals for classification.

XPASC measures to what degree a model, trained with a weakly labeled data set, can generalize from the information associated with the features, classes and labeling functions present in the data. Several considerations such as how important features[b] are for the classification and how much features, classes and labeling functions are correlated are taken into account. Therefore XPASC is composed of three parts: i) the explainability of each feature for a model, ii) its association with the class, and iii) its association with the labeling function. Note that both explainability and association contribute equally to XPASC. To calculate explainability, we use occlusion (Zeiler and Fergus 2014). The association strength is measured either with the PPMI or the CHI SQUARE-score. See Figure 1 for an illustration of XPASC.

3.1 Explainability

The term explainability expresses how important a single feature is for the classification of an instance, i.e., how the class prediction changes if the feature is omitted, masked or changed. To determine the importance of each feature of an instance for the classification task, we use the explainability method of occlusion. The idea for occlusion originally came from computer vision proposed by Zeiler and Fergus (2014), and since then it has been also used for Natural Language Processing, e.g., by Harbecke and Alt (2020), or Ancona et al. (2018).

[b] In this work, by features we mean the observable tokens of an input instance, in contrast to learned features from a network.

We perform occlusion in four steps as follows: we pass **i)** our input instance through our model and **ii)** the instance without the feature through the model. Explainability is then computed by **iii)** retrieving the prediction probabilities from both, and **iv)** calculating the Kullback-Leibler-Divergence (Kullback and Leibler 1951):

$$D_{KL}(P||Q) = \sum_{x \in X} P(x)\, log\left(\frac{P(x)}{Q(x)}\right) \tag{1}$$

where P and Q are discrete probability distributions and X is a shared probability space. By computing how different two probability distributions are, the Kullback-Leibler-Divergence indicates how much the occlusion affects the prediction in our case. Accordingly, we define the explainability of an instance and a feature as:

$$S_{xp}(i, f) = D_{KL}\left(P(i)||Q(i \setminus f)\right) \tag{2}$$

where $P(i)$ is the model prediction for the entire instance and $Q(i \setminus f)$ is the model prediction for the instance with feature f omitted. Note that both predictions are vectors of probability distributions over the set of possible classes. The smaller this difference of the two probability distributions, the less important the feature is for the classification result. In any case, the Kullback-Leibler-Divergence is between zero and one. The highest explainability is assigned to the features where the two prediction distributions differ the most, i.e., where the prediction changes greatly if the feature is omitted.

3.2 Association

Association measures the degree of correlation between observations. To find out how much a feature is correlated with its class and with its labeling function we calculate two association matrices with the following shapes:

$$|\mathbf{C}| = classes \times features$$

$$|\mathbf{L}| = labeling\_functions \times features$$

where $\mathbf{C}$ is the association matrix for classes and $\mathbf{L}$ is the matrix for labeling functions. The details for the matrix calculation are explained in section 3.2.1 and section 3.2.2. During the calculation of XPASC, the respective association value is looked up in the matrices for each feature given its class and its labeling function. To put both associations (feature and class/ feature and labeling function) in relation we subtract their scores and arrive with the overall association for a feature given its instance:

$$S_{asc}(i, f) = \sum_{j=1}^{N} \left(\mathbf{C}_{c^i f} - \mathbf{L}_{l^i_j f}\right) \tag{3}$$

where we iterate over all matching labeling functions l for instance i. Variable f denotes the feature, $\mathbf{C}_{c^i f}$ is the value of the association-matrix with respect to the class label of instance i, c_i, and $\mathbf{L}_{l^i_j f}$ the value for the j'th matching labeling function, l^i_y. By computing association that way the result can either be positive, negative or zero. Depending on the exact result, the score can then be interpreted directly. The larger (positive) S_{asc}, the more feature f is associated with the class, the smaller (negative) S_{asc}, the more feature f is associated with the labeling function. The closer the score is to zero, the more similar the association of the feature with the class and the labeling function.

Association can be modeled in different ways. We calculate it in two manners: using i) a chi squared test or ii) the positive pointwise mutual information.

3.2.1 CHI SQUARE*-based association*

For this option we calculate the association matrices based on univariate feature selection. This works by selecting the best features based on univariate statistical tests, in our case a chi squared (χ^2) test.

$$\text{CHI SQUARE}(z, f) = \frac{\left(o_{zf} - \tilde{o}_{zf}\right)^2}{\tilde{o}_{zf}} \tag{4}$$

where f is the feature, z represents a label (either class label or labeling function), o_{zf} is the absolute frequency of the combination of class/ labeling function and a feature (observed value) and $\tilde{o}_{zf}$ is the expected value of the absolute frequency of the combination of class / labeling function and a feature.

The χ^2 test measures the dependence between the features and the class/labeling function. Features with a high CHI SQUARE score are likely to be independent of the class/labeling function and therefore more irrelevant for the classification. The smaller the CHI SQUARE result, the more a feature is related to the class/labeling function and, consequently, the more important it is. Note that the CHI SQUARE association expresses which of the features are most associated with the class and include positive as well as negative correlation. Thus also negative examples can be found among the most associated ones, e.g. "brother" can have a very high association with the "married to" relation, although it is a negative indicator for that relation.

Formulas 5 and 6 define how to calculate one matrix entry for a feature and its corresponding class/ labeling function.

$$\mathbf{C}_{cf}^{\text{CHI SQUARE}} = \text{CHI SQUARE}(c, f) \tag{5}$$

$$\mathbf{L}_{lf}^{\text{CHI SQUARE}} = \text{CHI SQUARE}(l, f) \tag{6}$$

where c is the class, l the labeling function and f the feature.

3.2.2 PPMI*-based association*

As a second option we calculate the positive pointwise mutual information (Equation 8), where only the positive results of the pointwise mutual information (Equation 7) are taken into account. Assuming independence of two variables (in our case: a feature and a class/labeling function) PMI quantifies the discrepancy between the probability of their coincidence given their individual distributions and their joint distribution.

$$PMI(f, z) = log\left(\frac{P(f, z)}{P(f)P(z)}\right) \tag{7}$$

$$\text{PPMI}(f, z) = \begin{cases} PMI, \text{ if } PMI > 0 \\ 0 \qquad else \end{cases} \tag{8}$$

where f is a feature, z a label (either class label or labeling function), $P(f, z)$ the joint probability of a feature and a label, $P(f)$ the probability of the feature and $P(z)$ the probability of a label.

Formulas 9 and 10 define how to calculate one matrix entry for a feature and its corresponding class/ labeling function.

$$\mathbf{C}_{cf}^{\text{PPMI}} = \text{PPMI}(f, c) \tag{9}$$

$$\mathbf{L}_{lf}^{\text{PPMI}} = \text{PPMI}(f, l) \tag{10}$$

where c is the class, l the labeling function and f the feature.

3.3 The Combined Score: XPASC

XPASC (eXPlainability Association SCore) combines both explainability and association for one data set and a model. It measures how important each feature of an instance is and if it is more correlated with the class or with the labeling function. By multiplying the two measures and summing up over all instances and features we obtain:

$$S_{\text{XPASC}}(d, m) = 1 + \left(\frac{1}{N \times M} \sum_{i=1}^{N} \sum_{f=1}^{M} S_{xp}(i, f) \times S_{asc}(i, f) \right) \tag{11}$$

where d is the data set, m is the pre-trained model, N is the number of instances and M is the number of features. To make sure that XPASC is comparable across models and data sets we normalize by the data set size (number of instances times the number of features). Multiplying both components gives small negative values, so we add one to the result to make the final XPASC value above zero. The multiplication allows to put the importance of a feature in relation to its association with the class and the labeling function.

The sharpness of the explainability measure could be increased by a temperature hyperparameter γ for putting the focus only on the most relevant features ($\gamma \to \infty$) or equally on all features ($\gamma \to 0$), changing the XPASC formula as follows: $(S_{xp}(i, f)^{\gamma}) \times (S_{asc}(i, f))$. In this work, we considered the unchanged importance as given by the explainability algorithm ($\gamma = 1$).

Thus, a high XPASC indicates that many of the most important features (those that are effectively used for prediction) are correlated with the class instead of the labeling function. We can conclude that a high XPASC indicates more independence from the labeling functions and accordingly a greater generalization from the weak source. This also indicates a smaller degree of overfitting to the weak signals. Note that the results of CHI SQUARE-based XPASC and PPMI-based XPASC are scaled differently. This is due to their different characteristics, as well as the specific result values of the two calculations.

4. Weak Supervision Methods and Datasets

For our experiments we study four different weak supervision approaches, KNOWMAN, MAJORITY VOTE, SNORKEL-DP (Data Programming) (Ratner et al. 2020) and WEASEL (Cachay et al. 2021). Of those, only KNOWMAN provides an explicit mechanism for controlling the degree of generalization from the labeling functions. The latter three methods have been developed without any mechanism to control generalization.

Experiments are conducted for four classifications tasks: sentiment analysis, spam detection, detection of the spouse relationship and question classification. Four data sets that are common in weak supervision are used, which are SPAM, SPOUSE, IMDb and TREC, see Section 4.2.

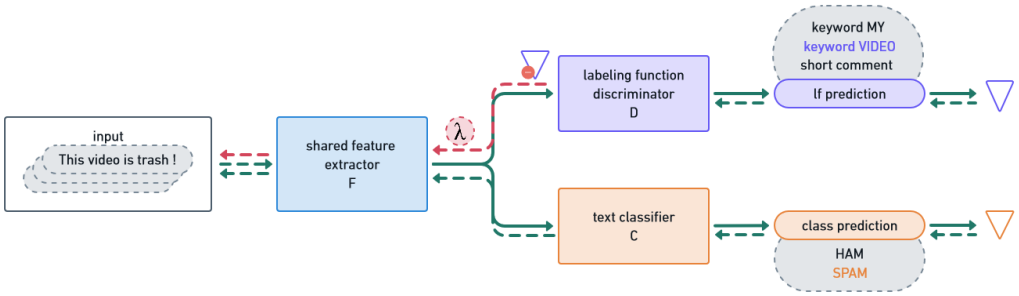


Figure 2. KNOWMAN architecture. One iteration for a pass of one batch of inputs. The correct class and labeling function for the example instance are highlighted. The parameters of $\mathcal{C}$ and $\mathcal{F}$ are updated together, the labeling function discriminator $\mathcal{D}$ is updated with a separate optimizer. Solid lines indicate forward, dashed lines the backward passes. ∇ indicates the (reversed) gradient. Before entering $\mathcal{F}$ the gradient is flipped. The influence of $\mathcal{D}$ can be controlled by hyperparameter λ. Strength of λ equals generalization strength.

4.1 Models

We use a pre-trained DistilBERT language model to encode the input texts. Similar to BERT (Devlin et al. 2019), DistilBERT is a masked transformer language model, which is a smaller, lighter, and faster version leveraging knowledge distillation while retaining 97% of BERT's language understanding capabilities (Sanh et al. 2019).

To arrive with the DistilBERT input encodings we first tokenize the texts using the DistilBERT tokenizer. After that, the tokenized input is converted to the DistilBERT transformer encoding, consisting of the input ids as well as the attention mask. We use that representation across KNOWMAN, SNORKEL-DP and MAJORITY VOTE models.

For WEASEL we encode the input using RoBERTa (Liu et al. 2019), a optimized version of the BERT language model, because DistilBERT is not supported by WEASEL.

4.1.1 KNOWMAN

In previous work we proposed KNOWMAN (März et al. 2021). The ultimate goal of KNOWMAN is to learn a feature representation that is invariant to the labeling functions which annotated the weakly supervised data. We showed that this representation is more general and more robust to incorrect classes that have been assigned by the labeling functions.

The architecture is designed as an adversarial model and contains three modules: i) a shared feature extractor $\mathcal{F}$, ii) a classifier $\mathcal{C}$ and iii) a labeling function discriminator $\mathcal{D}$. See Figure 2 for an illustration of the architecture. The classifier $\mathcal{C}$ is trained to predict the labels of a downstream task. The gradient of the loss function is used to optimize the classifier itself as well as the shared feature extractor. At the same time, the discriminator $\mathcal{D}$ is learned to distinguish between the different labeling functions and should predict which of the labeling functions was responsible for labeling an instance. The gradient of the discriminators loss function is used to optimize $\mathcal{D}$. In addition, the *reversed* gradient of $\mathcal{D}$ is used to optimize the feature extractor $\mathcal{F}$. This adversarial update leads to a weakening of the labeling function discrimination information and therefore to a better generalization. KNOWMAN uses a hyperparameter λ to control the level of weakening the signals. Consequently, XPASC allows us to study how changes in λ affect the degree of generalization of trained KNOWMAN models.

KNOWMAN is implemented as follows: the discriminator $\mathcal{D}$ is trained with a separate optimizer than $\mathcal{C}$ and $\mathcal{F}$. When $\mathcal{D}$ is trained, the parameters of $\mathcal{C}$ and $\mathcal{F}$ are frozen and vice versa. The losses for both, the classifier and the discriminator, are computed using negative log-likelihood (NLL). The classification NLL can be formalized as:

Table 1. KNOWMAN results on the test sets.

	SPAM	SPOUSE			IMDb
	Acc	P	R	F1	Acc
MAJORITY VOTE TF-IDF	0.87	0.12	**0.83**	0.20	0.65
BLIND KNOWMAN TF-IDF	0.91	0.12	0.76	0.21	0.75
SNORKEL-DP TF-IDF	0.81	**0.18**	0.63	0.28	0.50
KNOWMAN TF-IDF	**0.94**	0.16	0.72	**0.35**	**0.77**
Fine-tuned DistilBERT	**0.92**	0.14	0.78	0.24	0.70
MAJORITY VOTE DistilBERT	0.87	0.09	**0.90**	0.17	0.67
BLIND KNOWMAN DistilBERT	0.86	0.18	0.80	0.29	0.74
SNORKEL-DP DistilBERT	0.88	0.13	0.70	0.23	0.49
KNOWMAN DistilBERT	0.90	**0.27**	0.67	**0.39**	**0.76**

$$\mathcal{L}_{\mathcal{C}}(\hat{c}_i, c_i) = -\log P(\hat{c}_i = c_i) \tag{12}$$

where c_i is the (weakly supervised) annotated class and $\hat{c}_i$ is the prediction of the classifier module $\mathcal{C}$, for a training sample i. Analogously, the NLL for the labeling function discriminator is defined as:

$$\mathcal{L}_{\mathcal{D}}(\hat{l}_i, l_i) = -\log P(\hat{l}_i = l_i) \tag{13}$$

where l_i is the actual labeling function used for annotating sample i and $\hat{l}_i$ is the predicted labeling function by the discriminator $\mathcal{D}$.

The results of the experiments with KNOWMAN are shown in Table 1. As mentioned above, we encode the inputs with DistilBERT. In März et al. (2021) we did that for the experiments with KNOWMAN as well and additionally encoded the input with TF-IDF. Table 1 reports the results for experiments with both, DistilBERT and TF-IDF encodings. The baselines are a MAJORITY VOTE model as well a SNORKEL-DP model. The functionality of both models is explained in section 4.1.2. For DistilBERT encoded input we also trained a fine-tuned DistilBERT model and utilized it for prediction.

We refer to the KNOWMAN model with a λ value of zero as BLIND KNOWMAN. Setting λ to zero means disabling the generalization mechanism, because the feature extractor $\mathcal{F}$ is blind for the loss of the discriminator $\mathcal{D}$. KNOWMAN TF-IDF and KNOWMAN DistilBERT refer to a KNOWMAN model with optimal λ (tuned on the dev set through Bayesian hyperparameter optimization) for the respective dataset. KNOWMAN is able to outperform the baselines for all data sets. The only exception is fine-tuned DistilBERT, which performs better for SPAM.

4.1.2 Models without generalization mechanism

We also study three methods with no generalization control. This demonstrates that XPASC allows highlighting generalization across different approaches. The first method is majority voting. The second approach follows the data programming paradigm (DP) proposed by Ratner et al. (2016). The third approach (Cachay et al. 2021), WeaSEL, aims to learn in an end-to-end fashion from the labeling function output directly.

MAJORITY VOTE. For the majority vote classifier a matrix that holds the mapping between each labeling function and the corresponding class (the labeling function it is associated with) is needed. For each instance of the train set it is checked which labeling functions apply. Based

Table 2. Statistics on data set sizes. *Size after filtering* is the size of the data sets after instances without labeling function matches were filtered out.

Data set	original size	size after filtering	percentage of original size	labeling functions
SPAM	1586	1382	87.1	10
TREC	4965	4723	95.1%	68
SPOUSE	22254	5734	25.8%	9
IMDb	40000	39998	99.9%	6786

on this information it is looked up which class each matching labeling function would assign to the instance and if there is a majority for one class among them. If so, the class is assigned. If not, the class is either chosen at random from among the matching classes, or another special class is assigned to this instance. After labeling the training data in this way, we use the uncased DistilBERT model provided by Hugging Face (Wolf et al. 2019) as the prediction model.

SNORKEL-DP. We also compare to training models on labels denoised by SNORKEL-DP (Data Programming) (Ratner et al. 2020). To do this, we use Knodle's SNORKEL-DP wrapper (Sedova et al. 2021), where first a generative SNORKEL-DP model is learned, generating weak labels for the instances, and then a classification model (used for prediction) is trained with those labels. SNORKEL-DP works with a set of given labeling functions and learns a label model that focuses on the conflicts and agreements between the labeling functions to estimate their accuracy. For each labeling function an accuracy value is estimated to weigh their votes on each instance. Taking into account the weighted labeling functions, the label model can assign a probabilistic class to each instance and arrives with a weakly supervised data set. As with MAJORITY VOTE, the prediction model trained on the weak labels is uncased DistilBERT. The cross-entropy loss is optimzed on the probabilistic SNORKEL-DP labels.

WEASEL. In addition, we train models with WEASEL, an end-to-end model for weak supervision that does not take the noisy weak class labels, but the labeling function output as input for model training (Cachay et al. 2021). The approach produces accuracy scores for each labeling source (in our case labeling function) and trains both a neural encoder and a downstream model at the same time on the same loss by using each other's predicted labels as input. We use the WEASEL implementation of Zhang et al. (2021), train with the default hyperparameters and use RoBERTa as an encoder.

4.2 Data sets

For our experiments we use four standard data sets for weak supervision. In addition to the three binary data sets covered by KNOWMAN (SPAM, SPOUSE, IMDb) we also study one multi class data set (TREC). While SPAM, TREC and IMDb are classification tasks, SPOUSE addresses relation extraction.

4.2.1 SPAM

SNORKEL-DP provides a small subset of the YouTube comments data set (Alberto et al. 2015) where the task is to classify whether a text is relevant to a certain YouTube video or contains spam. Ten different labeling functions are used to assign the classes, mostly based on keywords and regular expressions. In contrast to other datasets, no development set is provided for SPAM, which is not relevant for XPASC but for downstream task training.

4.2.2 TREC

Another text classification data set is TREC which was proposed by (Li and Roth 2002) and addresses question classification. The data set contains automatically retrieved as well as manually constructed questions from six different classes. Multiple classes can be assigned for each instance, but the authors chose to design TREC as a single-class dataset. Therefore, the data set was manually annotated with one class per instance. However, the result of the initial overlapping of classes for each instance is that TREC is difficult to learn. We use the version of the data set provided by Zhang et al. (2021) within the WRENCH framework, containing 68 keyword-based labeling functions which have been generated by Awasthi et al. (2020).

4.2.3 SPOUSE

This data set addresses a binary relation extraction problem and aims to identify the *spouse* relation in text snippets. It has also been created by SNORKEL-DP and is based on the Signal Media One-Million News Articles Data set (Corney et al. 2016). The nine labeling functions use information from a knowledge base, keywords and patterns. One peculiarity of this data set is that it is very skewed, with over 90% of the instances not holding a spouse relation.

4.2.4 IMDb

The largest data set we use is IMDb, which contains movie reviews and is based on the data set from Maas et al. (2011). We use the IMDb version compiled by Sedova et al. (2021). All of the labeling functions used for this data set are occurrences of positive and negative keywords from Hu and Liu (2004). The addressed task for IMDb is binary sentiment analysis, classifying the reviews as either positive or negative. Unlike for the other two data sets, there are 6800 labeling functions for IMDb, which constitutes a particular challenge to the SNORKEL-DP denoising framework.

5. Experiments with XPASC

We present here the setup and findings of our analysis of different models, using XPASC. We evaluated XPASC both quantitatively across all models and with a qualitative feature analysis for KNOWMAN.

5.1 Evaluation settings

Since we want to calculate the correlations in XPASC on a representative amount of data for one data set, we use the train sets for the XPASC computations. Moreover, in a practical weak supervision setting (where XPASC might be used, e.g. for model selection), the existence of labeled development and test sets cannot be assumed, and the XPASC calculation needs to rely on weakly labeled training data only.

When using weak supervision to assign classes with labeling functions it can happen that instances do not have a labeling function match. Especially, for SPAM and SPOUSE many instances lack a labeling function match. Therefore, we filter out those instances with no labeling function matches for all our experiments and arrive with smaller data sets. For IMDb and TREC the number of instances does only change slightly, since there are very few instances without labeling function matches. Fortunately, the already very small SPAM data set does not get much smaller after filtering. For SPOUSE, we observe the greatest difference and only 25% of the original data set remain after filtering. See Table 2 for the data set sizes of the train sets.

In contrast to the results reported in März et al. (2021), we average results across 15 different seeds for SPAM and SPOUSE and across 5 different seeds for TREC to achieve more stable

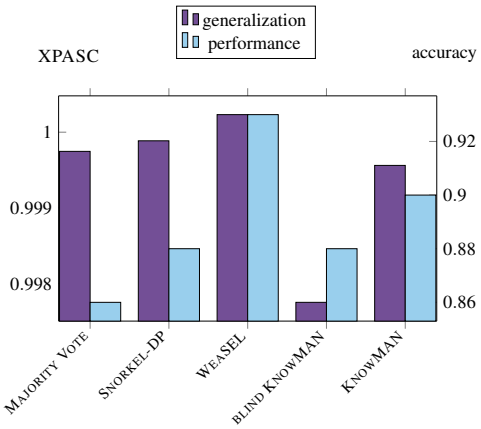


Figure 3. Evaluation of XPASC across baselines and KNOWMAN models for SPAM. BLIND KNOWMAN is a KNOWMAN model with λ set to 0. KNOWMAN for SPAM means a model trained with λ set to 2.

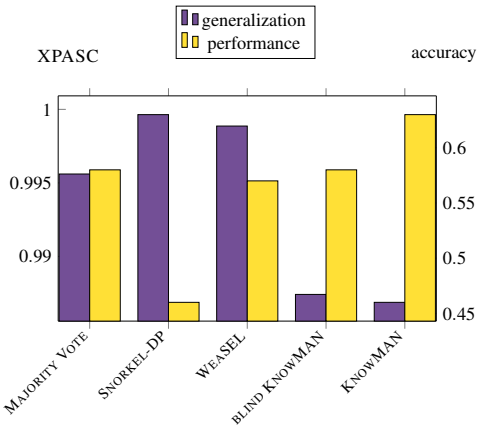


Figure 4. Evaluation of XPASC across baselines and KNOWMAN models for TREC. BLIND KNOWMAN is a KNOWMAN model with λ set to 0. KNOWMAN for TREC means a model trained with λ set to 0.001.

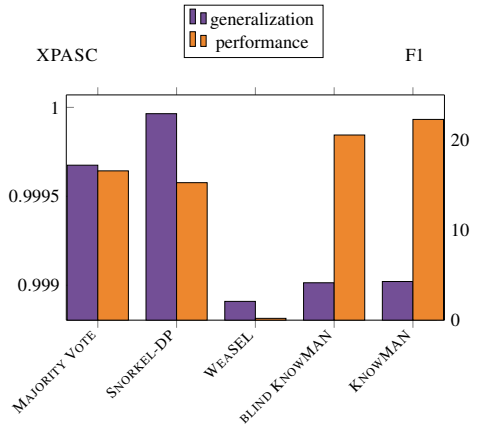


Figure 5. Evaluation of XPASC across baselines and KNOWMAN models for SPOUSE. BLIND KNOWMAN means a KNOWMAN model with λ set to 0. KNOWMAN for SPOUSE means a model trained with λ set to 0.5.

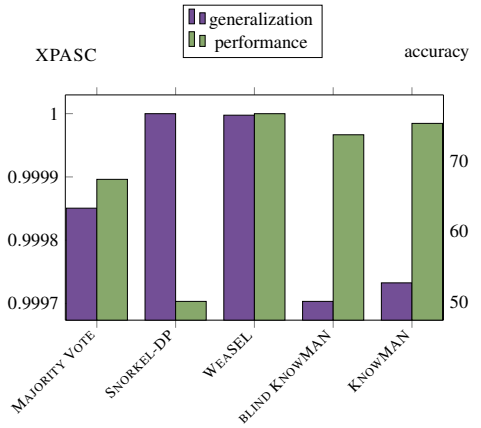


Figure 6. Evaluation of XPASC across baselines and KNOWMAN models for IMDb. BLIND KNOWMAN means a KNOWMAN model with λ set to 0. KNOWMAN for IMDb means a model trained with λ set to 0.5.

results. Due to the size of IMDb one XPASC run takes three days. To consume less resources, we performed one XPASC run with one seed for this data set only.

In our study of KNOWMAN with XPASC, we experiment with different values of λ. As the hyperparameter λ is intended to control the degree of generalization, this sheds light onto the the functionality of KNOWMAN. Specifically, it is useful for examining the hypothesis whether the model is able to generalize from the labeling functions when tuning λ. Our expectation is that the higher the value chosen for λ, the higher the XPASC result. With the experiments in this paper, we want to confirm that expectation and validate KNOWMAN with XPASC and vice versa. For MAJORITY VOTE, SNORKEL-DP and WEASEL we do not have a presumption of the XPASC result, but assume that the score could be higher than for KNOWMAN because these models have fewer parameters.

5.2 Quantitative evaluation

Our quantitative evaluation includes XPASC results across all models mentioned in section 4.1, as well as the results for different λ values for KNOWMAN. In addition, we evaluated KNOWMAN with XPASC for both association measures, CHI SQUARE and PPMI.

5.2.1 XPASC results across all models

We calculated XPASC with CHI SQUARE-based association for MAJORITY VOTE, SNORKEL-DP, WEASEL and KNOWMAN on all data sets.

The evaluation of SPAM (see Figure 3) shows the highest generalization for WEASEL and the SNORKEL-DP model achieves the second highest XPASC. Generalization scores for MAJORITY VOTE and the KNOWMAN model with the optimal λ value of 2 are similar and slightly worse than for SNORKEL-DP. The generalization of the BLIND KNOWMAN model with the disabled generalization mechanism is in contrast the lowest. The observations are different for the class prediction performance of the models. Here WEASEL and KNOWMAN give the highest, whereas MAJORITY VOTE gives the worst classification accuracy.

For TREC we observe the highest generalization for SNORKEL-DP and WEASEL (see Figure 4). Both achieve low classification numbers, although their performance differs significantly. SNORKEL-DP achieves the lowest results and WEASEL achieves results similar to BLIND KNOWMAN and MAJORITY VOTE. MAJORITY VOTE shows a similar classification performance as KNOWMAN but much higher generalization. The KNOWMAN models achieve the best classification performance, but have the lowest XPASC. Unlike for the other data sets, using KNOWMAN decreases the XPASC value slightly.

The evaluation of SPOUSE (see Figure 5) shows a picture similar to SPAM. Again, SNORKEL-DP achieves the highest XPASC. The scores of both KNOWMAN models are close to the generalization score of the WEASEL model. The difference between BLIND KNOWMAN and KNOWMAN is smaller than in the other data sets. In terms of performance, the models are ranked differently. Both KNOWMAN models perform better than SNORKEL-DP or MAJORITY VOTE. Indeed, the performance of SNORKEL-DP is the lowest, while this model has a high generalization value. The WEASEL model gives unreasonably low results for both, XPASC and F1 score, and like Stephan et al. (2022) we assume that this is due to the fact that they did not integrate large pre-trained language models like RoBERTa in their original work.

For IMDb the results are in agreement with the insights of the other data sets (see Figure 6). Again, SNORKEL-DP gives the highest XPASC and WEASEL the second highest XPASC result. Since BLIND KNOWMAN and KNOWMAN are only different by a λ value of 0.5, their generalization scores are close to each other. Again, MAJORITY VOTE reaches a slightly higher XPASC than the KNOWMAN models. The classification accuracy gives the same ranking as for SPOUSE, except for the WEASEL model, which performs best on the IMDb data set. Both KNOWMAN models perform better than SNORKEL-DP and MAJORITY VOTE, while having smaller XPASC results.

Overall we observe that models without explicit generalization modeling achieve higher XPASC values than KNOWMAN. This can be explained by the fact that these models employ a smaller number of layers, thus are less complex and enable greater generalization more easily. The more complex a model becomes and the more parameters it consists of, the more likely it is to overfit and thus generalization is more difficult to achieve. Another observation that can easily be drawn from the plots is that performance and generalization are not related one-to-one. Indeed, it seems like greater generalization often hinders performance. An explanation for this correlation may be that more generalization leads to less (over-)fitting, what can also harm performance. The works of Bartlett et al. (2019) and Zhang et al. (2020) show that models that overfit to noisy data still can achieve good performance and that the noise doesn't harm the performance

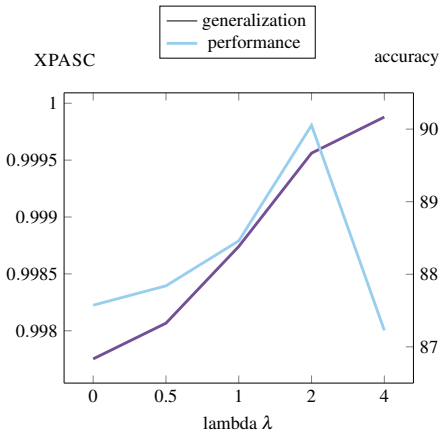


Figure 7. Chi-square-based XPASC and accuracy for SPAM across different λ values.

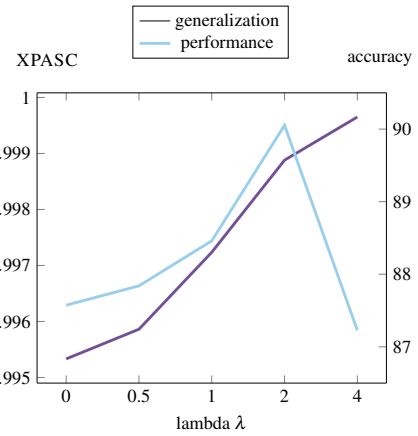


Figure 8. PPMI-based XPASC and accuracy for SPAM across different λ values.

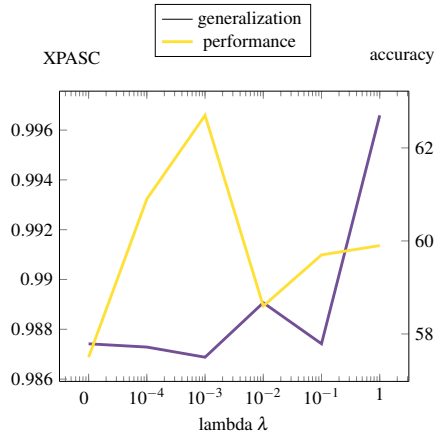


Figure 9. Chi-squared-based XPASC and accuracy for TREC across different λ values.

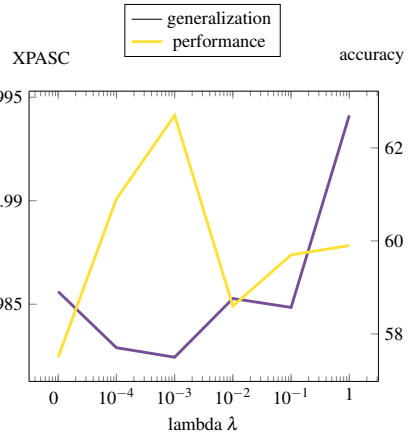


Figure 10. PPMI-based XPASC and accuracy for TREC across different λ values.

as much as expected. Still, Sanyal et al. (2021) claim that overfitting might not harm the performance of a model, but its robustness and that too much overfitting makes a model vulnerable (e.g. to adversarial attacks).

5.2.2 XPASC *results across* KNOWMAN *models*

To figure out if the degree of generalization can be controlled via the hyperparameter λ in the KNOWMAN models, we calculate XPASC for different λ values. We calculate both, XPASC CHI SQUARE and XPASC PPMI for SPAM, TREC and SPOUSE. For IMDb we calculate XPASC CHI SQUARE only, to save resources.

Figures 7 and 8 show the results for SPAM. It can be clearly seen that XPASC increases with increasing λ. The performance has its peak when using a λ value of 2.0. Both XPASC curves of the two association options are very similar in their shape, though, we find slightly smaller values for XPASC CHI SQUARE. To compare the two options for calculating association we draw scatter plots that depict the performance and generalization per run and seed (see Figures 14 and 15).

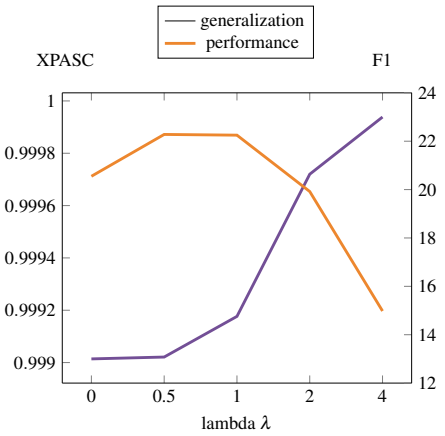


Figure 11. Chi-squared-based XPASC and F1 for SPOUSE across different λ values.

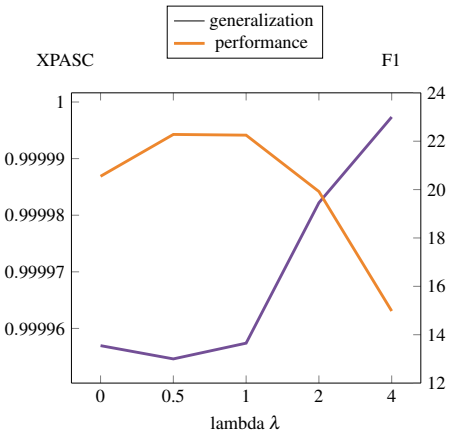


Figure 12. PPMI-based XPASC and accuracy for SPOUSE across different λ values.

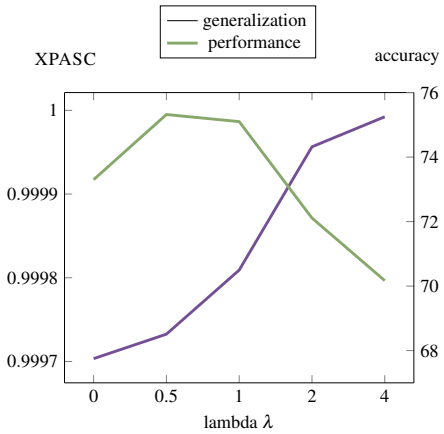


Figure 13. Chi-squared-based XPASC and accuracy for IMDb across different λ values.

For the CHI SQUARE option the accuracy and XPASC scores are clustered clearly recognizable for each λ value. For PPMI we observe slightly different results. Especially for lower λs the distributions are more mixed up. Still, both plots reflect the clustering of XPASC-values according to the chosen λ-values, and the performance trends, nicely.

The results for TREC (see Figures 9, 10) show that KNOWMAN is sensitive to small values of the hyperparameter λ in the multi-class setting. Using smaller λ values (in comparison to binary models) does improve the performance of the TREC model, whereas using greater λ values is less effective. The model with the best classification performance is trained with $\lambda = 0.001$. With regard to XPASC, one can clearly see from the curve that smaller λ values increase the generalization to a lesser extent than larger λ values. Moreover, the trends are less clear and effects are more brittle in this setting.

For SPOUSE, we observe a clear positive correlation of XPASC in relation to larger λ as well (see Figures 11, 12). Unlike for the SPAM and IMDb the curve is not strictly monotonically increasing, however. The scatter plots with the distributions of all results across the 15 runs show that for those λ values that cause the dips in the curve, there are some outliers that cause the lowerings (see Figures 16 and 17). In general SPOUSE appears more challenging and unstable,

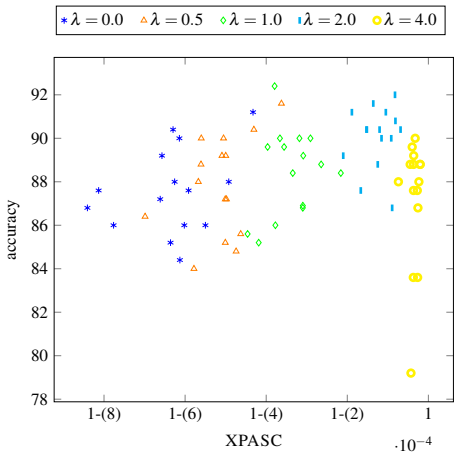


Figure 14. XPASC-CHI SQUARE and accuracy results of runs across 15 seeds for SPAM. Different colors indicate different λ values. Numbers in brackets must be multiplied by the subscript value, 10^{-4}.

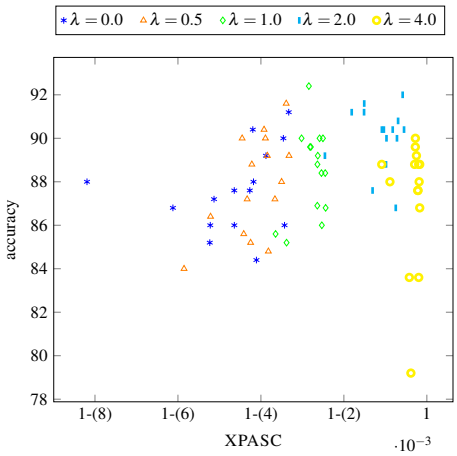


Figure 15. XPASC-PPMI and accuracy results of runs across 15 seeds for SPAM. Different colors indicate different λ values. Numbers in brackets must be multiplied by the subscript value, 10^{-3}.

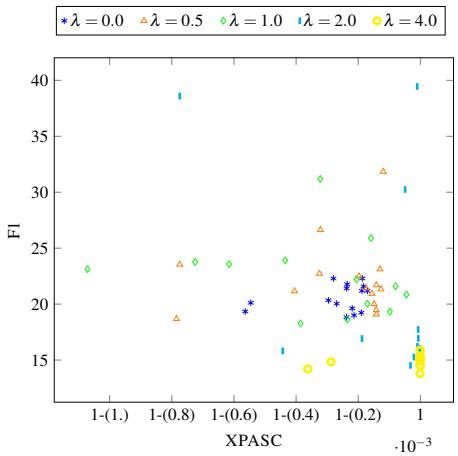


Figure 16. XPASC-CHI SQUARE and accuracy results of runs across 15 seeds for SPOUSE. Different colors indicate different λ values. Numbers in brackets must be multiplied by the subscript value, 10^{-3}.

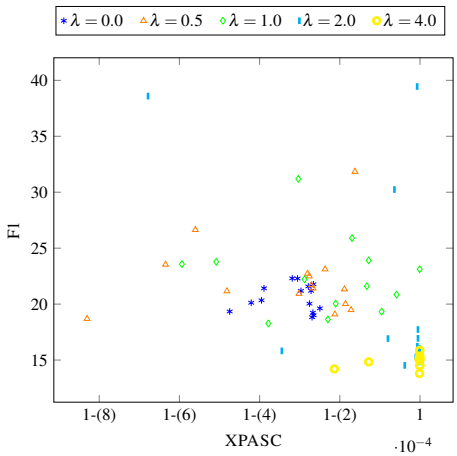


Figure 17. XPASC-PPMI and accuracy results of runs across 15 seeds for SPOUSE. Different colors indicate different λ values. Numbers in brackets must be multiplied by the subscript value, 10^{-4}.

having more outliers for both axes, XPASC and F1 score. In terms of performance, the optimum is reached with a λ value of 0.5 and decreases drastically with λ values being higher than 1.

The XPASC curve for IMDb increases monotonically in relation to λ (see Figure 13). There are no lowerings or peaks in the XPASC curve for this data set. As for SPOUSE, the best performance is reached with $\lambda = 0.5$ and decreases with λ being higher than 1.

Overall, we can conclude that it is possible to control the degree of generalization for the KNOWMAN models by using the respective hyperparameter λ. In addition, the results confirm our assumption that XPASC can reflect the generalization of models. As for the evaluation across all models, it shows that performance and generalization are not the same. The highest performance is not achieved with the greatest degree of generalization.

5.2.3 Magnitude of XPASC

The results show that the values for XPASC are very small. This is a consequence of composition of the formula: On the one hand, the CHI SQUARE and PPMI values are low (often zero or close to zero) already and the final association value (Formula 3) becomes even smaller due to the subtraction. On the other hand, the explainability values come from a probability distribution and therefore range between zero and one. By multiplying these small values, the results become even smaller. To bring the XPASC to a range with higher magnitude values, we experimented with the following normalization steps to calculate a scaled version of XPASC:

- scaling the range of the PPMI values between zero and one by using the normalized pointwise mutual information:

$$NPMI(f,z) = \frac{PMI(f,z)}{h(f,z)} \tag{14}$$

  where the pointwise mutual information in Equation 7 is normalized by $h(f,z)$, with $h(f,z)$ being the joint self-information $-log(P(f,z))$.
- scaling the explainability score to range between zero and one by normalizing the explainability of feature f given instance i by the maximum explainability value per instance:

$$S_{xp}(i,f) = \frac{D_{KL}\left(P(i)||Q(i \setminus f)\right)}{max\left(\{\forall f' \in i \mid D_{KL}\left(P(i)||Q(i \setminus f')\right)\}\right)} \tag{15}$$

  where $P(i)$ is the model prediction for the entire instance and $Q(i \setminus f)$ or $Q(i \setminus f')$ is the model prediction for the instance with feature f or f' omitted.
- scaling the distribution of both, explainability and association, to range between zero and one each by using MinMax scaling:

$$X_{std} = \frac{(X - X.min)}{(X.max - X.min)} \tag{16}$$

$$X_{scaled} = X_{std} \times (max - min) + min) \tag{17}$$

  where X is the array of all values (explainability or association) and min/max indicate the minimum/maximum value of the array.

This results in larger XPASC values, but normalizing and scaling the values can discard useful information. To investigate if there is an information loss due to the normalizing of the score, we compute another scatter plot that depicts the performance and generalization per run and seed for SPAM after applying the above mentioned steps to XPASC (see Figure 18). As one can clearly see, the individual characteristics are no longer as distinctly recognizable in the normalized version of XPASC (right plot) as they had been before (left plot). While the values for the individual KNOWMAN models used to be clearly separated from each other, the normalization has mixed them up. We conclude that important information is lost due to the normalization of the values. For this reason we accept the small values of the original formula in order to be able to optimally represent all information.

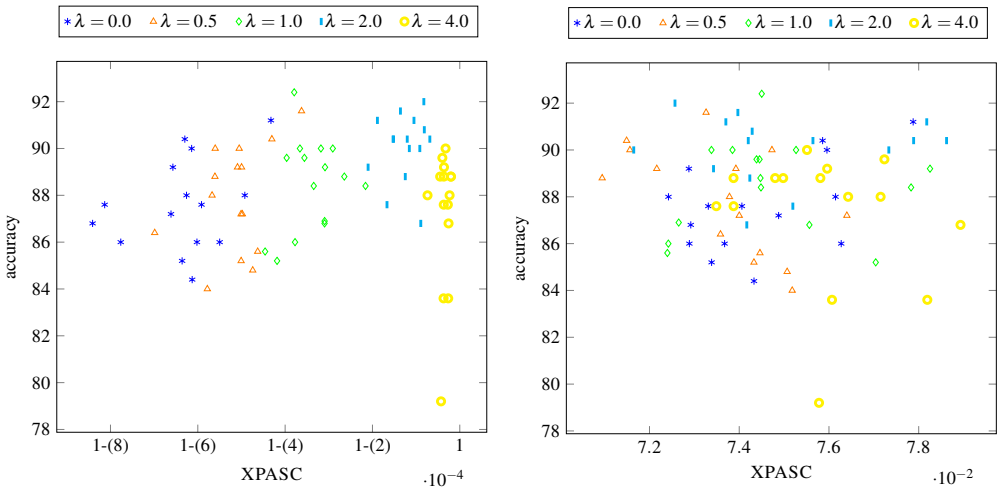


Figure 18. Original and **normalized** XPASC-CHI SQUARE and accuracy results of runs across 15 seeds for SPAM. Different colors indicate different λ values. Numbers in brackets must be multiplied by the subscript value, 10^{-4}.

5.3 Qualitative Feature Analysis

To confirm the functionality of XPASC qualitatively we took a close look at the data and their linguistic features to find instances that illustrate XPASC and its components.

First, we examined the association matrices to verify that they reflect the actual association of features, classes and labeling functions. See Tables 3 and 4 for the top five association values (for both CHI SQUARE and PPMI) of labeling functions and features for SPAM and SPOUSE. Because of the larger number of labeling functions we did not conduct this analysis for TREC and IMDb.

With regard to the CHI SQUARE-based association, the association between the labeling functions and the features can be understood easily. Many of the features most associated with their labeling function would also be considered very relevant by a human. In some cases, the features are even part of the pattern of the labeling function. For the PPMI-based association the top features contain parts of the patterns only rarely. Note, that PPMI is very sensitive to features that occur only once. These features obtain very high PPMI-based association scores because they are observed exclusively with a single class or labeling function. However, since these features all have a low frequency (occur only once), this is not a problem for the calculation of XPASC. Still, one can find words that are part of common expressions together with the keyword of the labeling function, e.g. “ALBUM” is associated with “my”. For SPOUSE one can find a lot of names among the PPMI-based association. To figure out if persons are married it is plausible that persons names are considered a lot.

The association of features with the classes is not as intuitive as for the labeling functions. For CHI SQUARE-based association the top ten features are identical for the classes for SPAM and almost identical for SPOUSE. Note that the CHI SQUARE-association only expresses which features were particularly relevant for determining the class. However, the correlation of these features with the class can be both positive (the feature is strongly associated because it gives a clue to the correct class) and negative (the feature is strongly associated because it gives a clue to distinguish it from other classes). The association with each class for the TREC data set is more intuitive in some cases. Words like “odor” or “malawi” are associated with the class “human being”, “cpr” or “p.m.” with “abbreviation”, what is plausible. On the other hand words like “make up” are associated with “location”. Since only one gold class could be chosen by the annotators

Table 3. Top 5 labeling function related features SPAM.

labeling function	**class**	**CHI SQUARE**	**PPMI**
keyword "my"	SPAM	my, channel, Hey, MY, probably	Cyphers, ALBUM, TOMORROW, READING, tvcmcadavid.weebly
keyword "subscribe"	SPAM	me, subscribe, to, subscribers, subscribe	Del, Rey, Drake, Macklemore, Pink
link	SPAM	V, \, STYLE, fight, 'http://youtu.be/9bZkp7q19f0'	scrubs, ./r, MontageParodies, AND, OTHER
keyword "Please"	SPAM	plz, please, Please, PLEASE, school	–, †,| , PLZZ, supporters
keyword "song"	HAM	song, This, songs, fun, Best	spare, upcoming, uk, rapper.please, worries
regex "Check out"	SPAM	out, this, on, Check, video	act, retain, delightful, system, rhythm
short comment	HAM	out, this, on, _, and	BR, sparkling-heart emoji, wonderful, LOST?, heart emoji
has person	HAM	Katy, Perry, Official, Charlie, Eminem	belle, chanson, lost?, clean, Eminem
polarity > 0.9	HAM	best, photo, Oppa, Yeah, Best	MOVES, MAKES, MEH, SMILE, EVER
subjectivity $>= 0.5$	HAM	only, views, YouTube:, love, out	Driveshaft, YEAH, Crazy, Flow, Ill

Table 4. Top 5 labeling function related features for SPOUSE.

labeling function	**class**	**CHI SQUARE**	**PPMI**
keyword "husband/wife"	SPOUSE	married, son, wife, husband, boyfriend	Peck, Veronique, glitters, Sweeting's, Body
keyword "husband/wife" left window	SPOUSE	written, wife, husband, conspiring, tunnel	Guys, Running, Role, wrestle, Off
same last name	SPOUSE	son, wife, husband, afternoon, daughter	Dudley, Hales, facilitating, Thomson, M&F
keyword "married"	SPOUSE	married, relationship, 2007., who, trainer	wakes, Gordon-Levitt, rowing, Regardless, minorities
familiar relationship	NO SPOUSE	son, wife, husband, sister, daughter	Fire, window, forcing, ribs, Claims
familiar relationship left window	NO SPOUSE	on, husband, half-brother, daughter, mother	hopelessness, hairdresser, sponsoring, Tailyour, Commandant
regex other relationship	NO SPOUSE	husband, boyfriend, planted, Gamble, David	Rita, Ora, Grimshaw, Zeinat, ordinary
known spouses from database	SPOUSE	Martin, denim, afternoon, Gwyneth, Paltrow	lacing, Amicable, exes, shifts, Basinger
last name known spouses	SPOUSE	Mara, mirror, tank, Paltrow, beauties	reported, Radar, spousal, grocery, head-on

Table 5. Top 10 class related features.

data set	class	CHI SQUARE	PPMI
SPAM	SPAM	subscribe, please, check, out, my, channel, song, Please, on, Check	beat, losing, ideas, lies, history, Driveshaft, YEAH, hot, Beats!!, STTUUPID
	HAM	subscribe, please, check, out, my, will, channel, song, Please, on	Drake, Macklemore, Pink, countless, inspire, FYI, freedom, speech, Lil, uploaded
TREC	description	cleaveland, cavaliers, monarchy, added, quisling, repossession, butcher, handful, spine, currency	per, chicken, dog, capital, university, black, island, san, south, west
	entity	sailed, talk-show, lends, surroundings, thalia, shakespearean, shylock, airforce, compiled, won-lost	leader, animals, words, father, christmas, held, nicknae, law, only, john
	human being	builds, resistance, odor, auh2o, mccain, rifleman, lai, malawi, zebulon, pike	god, square, mile, gas, strip, court, basketball, nationality, rock,month
	abbreviation	olympic, original, committee, aids, manufacturer, cpr, abbreviation, p.m., trinitrotoluene, equipment	monarchy, added, quisling, puerto, rico, repossession, butcher, handful, spine, 's
	location	aborigines, adventours, tours, photosynthesis, makeup, erykah, badu, m, ayer, bend	said, so, letter, kennedy, bridge, human, nixon, no, river, his
	numeric value	56-game, streak, graffiti, quilting, iran-contra, deere, tractors, cherubs, puerto, rico	square, strip, court, jackson, basketball, numbers, university, john, show
SPOUSE	SPOUSE	married, son, wife, husband, boyfriend, , young, family, sister, daughter	ringing, Sweeting's, Body, Cutting, Crap, Australians, marches, splashes, Kingi
	NO SPOUSE	married, son, wife, husband, boyfriend, , young, family, younger, sister	ordinary, rank-and-file, handpicked, Abu, Bakr, al-Baghdadi, L.L., J.'s, trendy
IMDB	negative	Instead, back, character, too, much, does, entire, cast, So, bizarre	Zwarts, Fredrikstad, Hilarios!10, wawa, CONSIDERING, Hobb's, Smooch, Investigative, belly-dancers, retirony
	positive	writing, It's, most, drag, you, us, won, Oscar, those, endless	Culpability, Package, slip-ups, AARP, Symona, Boniface, Lorch, Lynton, Tyrrell, Heinie

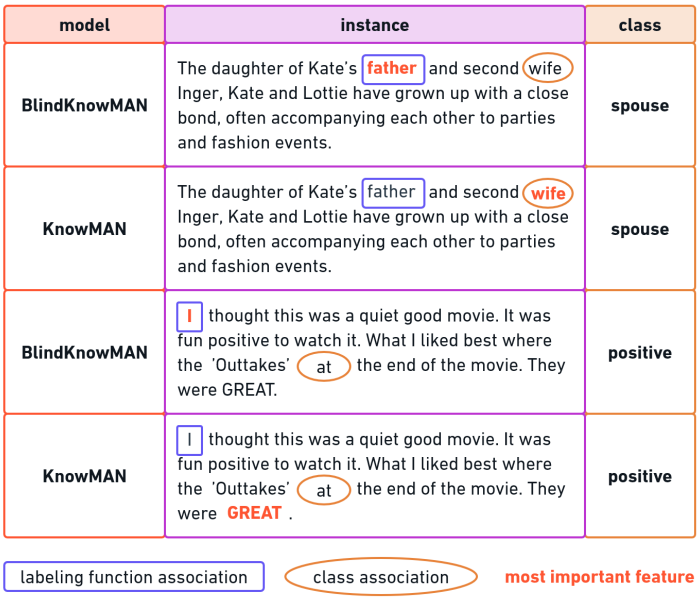


Figure 19. Examples from SPOUSE and IMDb where the feature with the highest explainability is shifted from a misleading labeling function towards the correct class. Association is based on CHI SQUARE and KNOWMAN uses $\lambda = 4$.

during the labeling of the data set, it is likely that some words would also be suitable for another class that had been in the set of suitable classes associated with the instance. For IMDb the features are different for both classes and very intuitive for a human. Features like "bizarre" or "Oscar" clearly point to a certain sentiment.

For PPMI-based association the top ten features are different for all data sets. Again this ranking is not easily interpretable for a human, but reflects the association and co-occurrence in the weakly supervised data sets. Because of the sensitivity to rare words, we found many features with the same association score, and the top n features in Table 5 are therefore only an excerpt.

Next, we looked for instances that confirm the functionality of XPASC and KNOWMAN. Therefore, we compared BLIND KNOWMAN and KNOWMAN with $\lambda = 4$. The ultimate and most challenging requirement for the models would be the following: Shift the focus from features that are associated with a deceptive labeling function towards features that are associated with the correct class. Two kinds of information need to be found for that goal: i) a feature that is very important for the classification (has a very high explainability score) and is associated most with a misleading labeling function, pointing to the wrong class, in BLIND KNOWMAN and ii) the KNOWMAN model is able to shift the highest explainability to a feature that is not associated with the misleading labeling function anymore, but with the correct class. Thus in the KNOWMAN model the most important feature should be associated with the correct class most. For example in Figure 19 the first instance should be classified as holding a *spouse* relation. The most important feature for BLIND KNOWMAN is **father**, what actually would lead to the classification of *no spouse*. The KNOWMAN model achieves the shift to the feature **wife**, that is a better indicator for the spouse relation. The second example in Figure 19 is drawn from IMDb. The review is *positive* and the feature 'I' is associated with the negative class. KNOWMAN manages to shift the focus to the feature 'GREAT', what is a better indicator for a positive sentiment.

We also measured this shift of the most important feature quantitatively. For SPAM, we found a shift is achieved for 12 instances and for SPOUSE 267 instances (both on average across seeds).

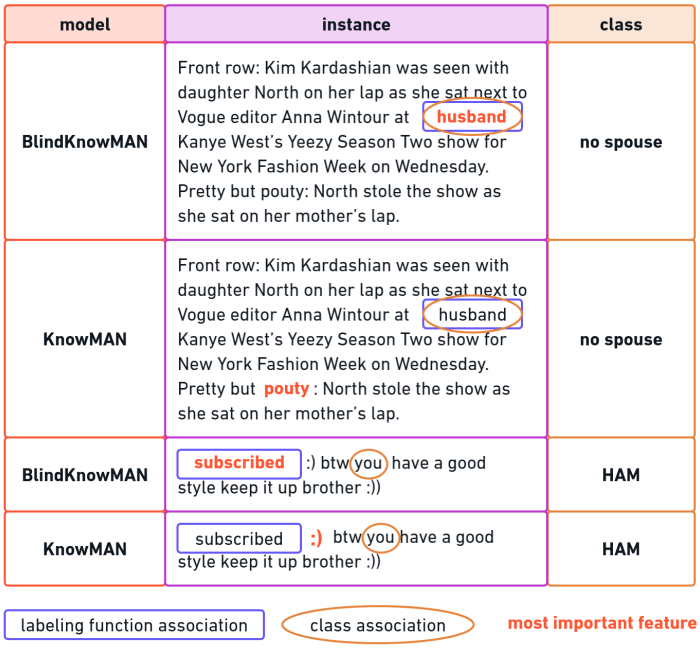


Figure 20. Examples from SPAM and SPOUSE. Highest explainability is shifted from features that are associated with the labeling function to other features. Association is based on CHI SQUARE and KNOWMAN uses $\lambda = 4$.

For IMDb the model manages to shift the misleading feature to the correct one for 7 instances. This comparison always refers to BLIND KNOWMAN and KNOWMAN with $\lambda = 4$.

A better generalization can also be achieved if the focus is shifted from the misleading feature to another feature that is not associated with the correct class, but at least is no longer associated with the labeling function. See Figure 20 for examples that illustrate this. The first example, again from SPOUSE, expresses a *no spouse* relation, but the most important feature for BLIND KNOWMAN is **husband**. Shifting the focus to another word - **pouty** - KNOWMAN is able to assign the correct label. The second example is comes from SPAM, where BLIND KNOWMAN considers the most important feature as **subscribed** for an instance that actual belongs to the *HAM* class. Since this is misleading, KNOWMAN focuses on the emoticon in the instance and assigns the correct label.

In addition, we noticed in the linguistic feature-based analysis that the weak labels for SPOUSE are very noisy and imprecise. We found many instances where a human annotator would have assigned another class than the labeling functions assigned.

Overall, the quantitative results can confirm our findings of the qualitative analysis. The KNOWMAN architecture is able to increase generalization and XPASC is a good indicator for the generalization ability of models.

6. Conclusion

We presented XPASC, a novel score to measure generalization for weakly supervised models. Our extensive analysis shows that XPASC is able to reflect the generalization of models given a dataset and the labeling functions used to perform weak supervision. In addition, we studied the adversarial approach KNOWMAN, designed to enable the control of generalization in weakly supervised models. We confirmed the hypothesis that the architecture is able to control the shift

from labeling functions to other signals by a hyperparameter. We also showed that performance and generalization do not relate one-to-one and it has to be decided based on the task, dataset and model, which degree of generalization is desired. XPASC can be used with any pre-trained weakly supervised model, a dataset and its set of applied labeling functions. Assuming that many neural models, designed to work with noisy weakly supervised data, are complex and thus suffer from overfitting, XPASC can serve as an indicator for their ability to fit unseen data. In general the core components of XPASC, explainability and association, are interchangeable, what makes the score flexible in practice.

Competing interests declaration

Competing interests: The authors declare none.

Acknowledgements

This research was funded by the WWTF through the project "Knowledge-infused Deep Learning for Natural Language Processing" (WWTF Vienna Research Group VRG19-008).

References

Alberto, T. C., Lochter, J. V., and Almeida, T. A. 2015. Tubespam: Comment spam filtering on youtube. In **Li, T., Kurgan, L. A., Palade, V., Goebel, R., Holzinger, A., Verspoor, K., and Wani, M. A.**, editors, *14th IEEE International Conference on Machine Learning and Applications, ICMLA 2015, Miami, FL, USA, December 9-11, 2015*, pp. 138–143. IEEE.

Ancona, M., Ceolini, E., Öztireli, C., and Gross, M. 2018. Towards better understanding of gradient-based attribution methods for deep neural networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

Awasthi, A., Ghosh, S., Goyal, R., and Sarawagi, S. 2020. Learning from rules generalizing labeled exemplars. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Bartlett, P. L., Long, P. M., Lugosi, G., and Tsigler, A. 2019. Benign overfitting in linear regression. *CoRR*, abs/1906.11300.

Cachay, S. R., Boecking, B., and Dubrawski, A. 2021. End-to-end weak supervision. In **Ranzato, M., Beygelzimer, A., Dauphin, Y. N., Liang, P., and Vaughan, J. W.**, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pp. 1845–1857.

Corney, D. P. A., Albakour, D., Martinez-Alvarez, M., and Moussa, S. 2016. What do a million news articles look like? In **Martinez-Alvarez, M., Kruschwitz, U., Kazai, G., Hopfgartner, F., Corney, D. P. A., Campos, R., and Albakour, D.**, editors, *Proceedings of the First International Workshop on Recent Trends in News Information Retrieval co-located with 38th European Conference on Information Retrieval (ECIR 2016), Padua, Italy, March 20, 2016*, volume 1568 of *CEUR Workshop Proceedings*, pp. 42–47. CEUR-WS.org.

Dehghani, M., Severyn, A., Rothe, S., and Kamps, J. 2017. Avoiding your teacher's mistakes: Training neural networks with controlled weak supervision. *CoRR*, abs/1711.00313.

Devlin, J., Chang, M., Lee, K., and Toutanova, K. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In **Burstein, J., Doran, C., and Solorio, T.**, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pp. 4171–4186. Association for Computational Linguistics.

Fu, D. Y., Chen, M. F., Sala, F., Hooper, S. M., Fatahalian, K., and Ré, C. 2020. Fast and three-rious: Speeding up weak supervision with triplet methods. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pp. 3280–3291. PMLR.

Harbecke, D. and Alt, C. 2020. Considering likelihood in NLP classification explanations with occlusion and language modeling. In **Rijhwani, S., Liu, J., Wang, Y., and Dror, R.**, editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop, ACL 2020, Online, July 5-10, 2020*, pp. 111–117. Association for Computational Linguistics.

Hsieh, C., Zhang, J., and Ratner, A. 2022. Nemo: Guiding and contextualizing weak supervision for interactive data programming. *CoRR*, abs/2203.01382.

Hu, M. and Liu, B. 2004. Mining and summarizing customer reviews. In **Kim, W., Kohavi, R., Gehrke, J., and DuMouchel,**

W., editors, *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Seattle, Washington, USA, August 22-25, 2004*, pp. 168–177. ACM.

Kullback, S. and Leibler, R. A. 1951. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86.

Li, H., **Li, J.**, **Guan, X.**, **Liang, B.**, **Lai, Y.**, **and Luo, X.** 2019. Research on overfitting of deep learning. In *15th International Conference on Computational Intelligence and Security, CIS 2019, Macao, SAR, China, December 13-16, 2019*, pp. 78–81. IEEE.

Li, X. and Roth, D. 2002. Learning question classifiers. In *19th International Conference on Computational Linguistics, COLING 2002, Howard International House and Academia Sinica, Taipei, Taiwan, August 24 - September 1, 2002*.

Liu, Y., **Ott, M.**, **Goyal, N.**, **Du, J.**, **Joshi, M.**, **Chen, D.**, **Levy, O.**, **Lewis, M.**, **Zettlemoyer, L.**, **and Stoyanov, V.** 2019. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

Maas, A. L., **Daly, R. E.**, **Pham, P. T.**, **Huang, D.**, **Ng, A. Y.**, **and Potts, C.** 2011. Learning word vectors for sentiment analysis. In **Lin, D.**, **Matsumoto, Y.**, **and Mihalcea, R.**, editors, *The 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Proceedings of the Conference, 19-24 June, 2011, Portland, Oregon, USA*, pp. 142–150. The Association for Computer Linguistics.

März, L., **Asgari, E.**, **Braune, F.**, **Zimmermann, F.**, **and Roth, B.** 2021. Knowman: Weakly supervised multinomial adversarial networks. In **Moens, M.**, **Huang, X.**, **Specia, L.**, **and Yih, S. W.**, editors, *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pp. 9549–9557. Association for Computational Linguistics.

Mauri, M., **Gili, K.**, **and Perdomo-Ortiz, A.** 2022. Evaluating generalization in classical and quantum generative machine learning models: Part i. *Bulletin of the American Physical Society*.

Power, A., **Burda, Y.**, **Edwards, H.**, **Babuschkin, I.**, **and Misra, V.** 2022. Grokking: Generalization beyond overfitting on small algorithmic datasets. *CoRR*, abs/2201.02177.

Ratner, A., **Bach, S. H.**, **Ehrenberg, H. R.**, **Fries, J. A.**, **Wu, S.**, **and Ré, C.** 2020. Snorkel: rapid training data creation with weak supervision. *VLDB J.*, 29(2-3):709–730.

Ratner, A., **Hancock, B.**, **Dunnmon, J.**, **Sala, F.**, **Pandey, S.**, **and Ré, C.** 2019. Training complex models with multi-task weak supervision. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pp. 4763–4771. AAAI Press.

Ratner, A. J., **Sa, C. D.**, **Wu, S.**, **Selsam, D.**, **and Ré, C.** 2016. Data programming: Creating large training sets, quickly. In **Lee, D. D.**, **Sugiyama, M.**, **von Luxburg, U.**, **Guyon, I.**, **and Garnett, R.**, editors, *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pp. 3567–3575.

Ren, W., **Li, Y.**, **Su, H.**, **Kartchner, D.**, **Mitchell, C. S.**, **and Zhang, C.** 2020. Denoising multi-source weak supervision for neural text classification. In **Cohn, T.**, **He, Y.**, **and Liu, Y.**, editors, *Findings of the Association for Computational Linguistics: EMNLP 2020, Online Event, 16-20 November 2020*, volume EMNLP 2020 of *Findings of ACL*, pp. 3739–3754. Association for Computational Linguistics.

Robinson, J., **Jegelka, S.**, **and Sra, S.** 2020. Strength from weakness: Fast learning using weak supervision. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pp. 8127–8136. PMLR.

Roelofs, R., **Shankar, V.**, **Recht, B.**, **Fridovich-Keil, S.**, **Hardt, M.**, **Miller, J.**, **and Schmidt, L.** 2019. A meta-analysis of overfitting in machine learning. In **Wallach, H. M.**, **Larochelle, H.**, **Beygelzimer, A.**, **d'Alché-Buc, F.**, **Fox, E. B.**, **and Garnett, R.**, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 9175–9185.

Salman, S. and Liu, X. 2019. Overfitting mechanism and avoidance in deep neural networks. *CoRR*, abs/1901.06566.

Sanh, V., **Debut, L.**, **Chaumond, J.**, **and Wolf, T.** 2019. Distilbert, a distilled version of BERT: smaller, faster, cheaper and lighter. *CoRR*, abs/1910.01108.

Sanyal, A., **Dokania, P. K.**, **Kanade, V.**, **and Torr, P. H. S.** 2021. How benign is benign overfitting ? In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Sedova, A., **Stephan, A.**, **Speranskaya, M.**, **and Roth, B.** 2021. Knodle: Modular weakly supervised learning with pytorch. In **Rogers, A.**, **Calixto, I.**, **Vulic, I.**, **Saphra, N.**, **Kassner, N.**, **Camburu, O.**, **Bansal, T.**, **and Shwartz, V.**, editors, *Proceedings of the 6th Workshop on Representation Learning for NLP, RepL4NLP@ACL-IJCNLP 2021, Online, August 6, 2021*, pp. 100–111. Association for Computational Linguistics.

Stephan, A., **Kougia, V.**, **and Roth, B.** 2022. Sepll: Separating latent class labels from weak supervision noise. *CoRR*, abs/2210.13898.

Sukhbaatar, S., **Bruna, J.**, **Paluri, M.**, **Bourdev, L. D.**, **and Fergus, R.** 2014. Training convolutional networks with noisy labels. *arXiv: Computer Vision and Pattern Recognition*.

Wolf, T., **Debut, L.**, **Sanh, V.**, **Chaumond, J.**, **Delangue, C.**, **Moi, A.**, **Cistac, P.**, **Rault, T.**, **Louf, R.**, **Funtowicz, M.**, **and Brew, J.** 2019. Huggingface's transformers: State-of-the-art natural language processing. *CoRR*, abs/1910.03771.

Yu, Y., **Zuo, S.**, **Jiang, H.**, **Ren, W.**, **Zhao, T.**, **and Zhang, C.** 2021. Fine-tuning pre-trained language model with

weak supervision: A contrastive-regularized self-training approach. In **Toutanova, K.**, **Rumshisky, A.**, **Zettlemoyer, L.**, **Hakkani-Tür, D.**, **Beltagy, I.**, **Bethard, S.**, **Cotterell, R.**, **Chakraborty, T.**, **and Zhou, Y.**, editors, *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*, pp. 1063–1077. Association for Computational Linguistics.

Zeiler, M. D. and Fergus, R. 2014. Visualizing and understanding convolutional networks. In **Fleet, D. J.**, **Pajdla, T.**, **Schiele, B.**, **and Tuytelaars, T.**, editors, *Computer Vision - ECCV 2014 - 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part I*, volume 8689 of *Lecture Notes in Computer Science*, pp. 818–833. Springer.

Zhang, J., **Hsieh, C.**, **Yu, Y.**, **Zhang, C.**, **and Ratner, A.** 2022a. A survey on programmatic weak supervision. *CoRR*, abs/2202.05433.

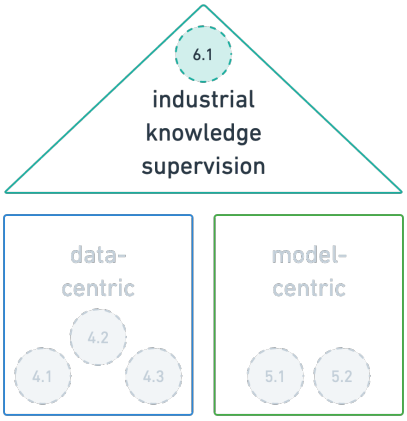
Zhang, J., **Wang, H.**, **Hsieh, C.**, **and Ratner, A.** 2022b. Understanding programmatic weak supervision via source-aware influence function. *CoRR*, abs/2205.12879.

Zhang, J., **Yu, Y.**, **NameError**, **Wang, Y.**, **Yang, Y.**, **Yang, M.**, **and Ratner, A.** 2021. WRENCH: A comprehensive benchmark for weak supervision. In **Vanschoren, J. and Yeung, S.**, editors, *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual*.

Zhang, M., **Fujinuma, Y.**, **Paul, M. J.**, **and Boyd-Graber, J. L.** 2020. Why overfitting isn't always bad: Retrofitting cross-lingual word embeddings to dictionaries. In **Jurafsky, D.**, **Chai, J.**, **Schluter, N.**, **and Tetreault, J. R.**, editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pp. 2214–2220. Association for Computational Linguistics.

6. Knowledge Supervision for Industrial Information Extraction

6.1. Turning a Rule-based System into a Machine Learning Model



Contributing Article

Submitted to *European Chapter of the Association for Computational Linguistics (EACL) 2023* on 20 October 2022.

März, L., Altergott, C., Stephan, A., & Roth, B. (2022). Recycle your Rules: How to turn a Rule-based System into a Machine Learning Model.

Author Contributions

The conception of the work was developed by Luisa März and Benjamin Roth. In addition, Christoph Ringlstetter contributed ideas. Programming was mainly done by Luisa März and Christian Altergott provided support for software engineering problems. The parts Luisa März programmed include data compilation, data cleaning, anonymization, and pre-processing, parsing the output of the rule-based system, weak supervision code, annotation of data with weak supervision, PyTorch-based model training, as well as scripts for prediction, evaluation, and analysis. Christian Altergott supported with code on data pre-processing and information extraction from the rule-based system. The writing of the manuscript was mainly done by Luisa März with large contributions in revising and rewriting by Andres Stephan. Besides the authors, Vasiliki Kougia also contributed feedback on the manuscript. Luisa März, Andreas Stephan, and Benjamin Roth proofread the final version of the manuscript for submission.

Recycle your Rules: How to turn a Rule-based System into a Machine Learning Model

Anonymous EACL submission

Abstract

Rule-based systems for data processing are often still the norm in many industrial contexts, capturing important knowledge about business processes that need to be automated. In this article, we establish best practices and present a step-by-step guide on how to transform rule-based systems into machine learning models from a weak supervision perspective. Programmatic weak supervision requires labeling functions that associate configurations in the data with a desired output. We describe in detail how to generate labeling functions from existing systems and how to use them to annotate data. In addition to a high-level perspective that contains fundamentals of the method on a more abstract level, we illustrate the process using an example from an automotive company and carry out experiments with the weakly supervised data. In this way, we want to enable practitioners to apply the process in their context. We point out difficulties and stumbling blocks along the way and describe steps how to deal with them. Our contributions include: i) a structured guide on how to leverage rule-based systems with the help of weak supervision and ii) an example from the automotive domain, including a detailed evaluation, comparing a rule-based system to established standard and weak supervision taggers for Named Entity Recognition.

1 Introduction

Machine learning and deep learning models have increasingly found their way into industry. The use cases in industry are often very specific (in terms of tasks, data, and complexity) and deviate substantially from the standard settings studied in academic research. Companies need therefore to either solve a task using rule-based approaches or train their own machine learning models, which requires labeled data. Manual labeling is an expensive and inflexible approach to create labels. However, there is a variety of strategies for automated labeling and one of the most popular strategies is weak supervision. Weak supervision aims to annotate large amounts of data by using heuristics and patterns. Templates to label data are called labeling functions (LFs) in this context. A LF can be a rule, regular expression, keyword match, or other heuristic that allows a label to be assigned automatically. In this way, a large labeled data set can be obtained quickly and with noticeably less manual effort. Nevertheless, these heuristics and patterns must be defined first.

Currently, many companies actively use rule-based algorithms. These algorithms often perform sufficiently well, but tend to be complex pipelines that are expensive to develop and maintain. Moreover, rule-based systems can only capture what has been explicitly specified, and apart from that they have no generalization abilities. On the other hand, not using the knowledge contained in already existing rule-based systems would be a waste of resources. For this reason, it makes sense to take advantage of existing rule-based systems and view them in the light of weak supervision. This way large amounts of annotated data can be retrieved while still using existing systems instead of discarding them entirely. In addition, the neural weak supervision models are easier to use and more flexible, both when compared to rule-based systems (in terms of generalization abilities) and to systems based on annotated data (in terms of dealing with changing task specifications).

In this work, we present a step-by-step guide on how to transform a rule-based system into a machine learning system by leveraging weak supervision. We aim to provide a practical and understandable guide for companies to recycle their rule-based systems for modern machine learning approaches. We will discuss how, ideally, weak supervision can be used to label data at scale in an organization and then train a machine learning model on that labeled data. The focus here is par-

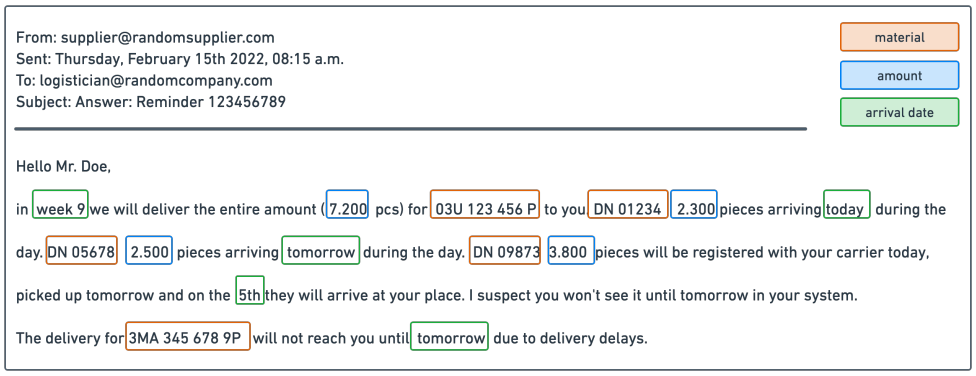


Figure 1: Example supplier email with marked material, amount and arrival date.

ticularly on how to develop and use LFs from a rule-based system.

In order to make the process more tangible, we will use an example from industry as a guide in this work and illustrate the individual steps using this example. At the same time, our goal is to keep a high-level perspective in order to describe the process in general terms. The following real-world example, which is very typical for the industrial context and comes from the automotive domain, runs through the remainder of the paper:

In a company, deliveries from different suppliers have to be coordinated by logisticians and the corresponding information needs to be transferred to a database (that can then be accessed by an enterprise application software). Not all supplier interactions can be captured programmatically, and in reality a lot of important communication is done via emails, that then need to be retroactively inserted into the databases. Hence, the information relevant to the database needs to be extracted from the text of the emails. Typical information that should be extracted is mentioned **material**, **amount** of the material and **time of departure** and/or **arrival** (denoted by **MAT**, **PC**, **ETA** and **ETD** labels). See Figure 1 for an example sentence. We compare different neural sequence tagging models to extract the desired information automatically: standard sequence tagging architectures, as well as an adversarial architecture and two label models for weak supervision.

Our contributions are: i) a structured guide on how to leverage rule-based systems with the help of weak supervision and ii) a sequence tagging example from the automotive domain illustrating the process and including a detailed evaluation of comparing a rule-based system to established standard and weak supervision models for Named Entity Recognition.

The remainder of the paper contains the detailed description about the steps of the system conversion in Section 2 and the sequence tagging experiments with real-world data from supplier emails as well as the results and analysis in Section 3. We also discuss related work in the field of weak supervision and the application of weak supervision in industry in Section 4.

2 System conversion: From Rule-based to Machine Learning

We propose the following steps to turn the existing rule-based system to a machine learning system: i) machine learning problem determination, ii) data collection, iii) labeling function identification, iv) weak supervision. All the steps are explained in detail in the following sections. As a running example, we use email communication within a company's logistics, whereby information has to be extracted from the emails. Within our case study in section 2 we go into this in detail.

2.1 Machine Learning Problem Determination

The first step is to find out if and how the task to be solved can be viewed as a machine learning problem. The second step is to identify what type of machine learning problem best represents the task.

In our running example, every token should be categorized as containing a certain information piece or not. Thus we can solve this problem with

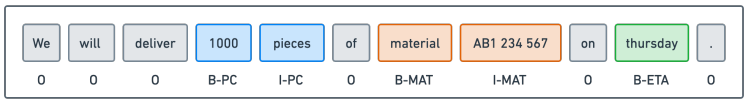


Figure 2: Example sentence from the labeled corpus with custom named entities in the IOB2 format.

sequence tagging. See Figure 2 for a tagged example sentence.

Suitable machine learning tasks for other problems could be classification, e.g., to which kind of email does the entire email belong (spam, ham etc.?) or sentiment analysis (is the tone in the email neutral, positive or negative?). Once the machine learning problem is identified the input and output of the model need to be determined. In the machine learning context the information is encoded in input representations and labels and can be formalized as follows. Let $X = (x_1, ..., x_n)$ be the collection of unlabeled data points and $Y = (y_1, ..., y_n)$ the labels to be determined. Note that the set of labels $C, y_i \in C$ must also be specified in advance.

In our case the labels are custom entities (e.g., **MAT**, **ETA**). One entity can either be just a single token (e.g., "1000") or it can be a span of entities and consist of multiple tokens (e.g., "1000 pieces"), see also Figure 2. Hence, the model output follows the IOB-style tagging scheme. See Alshammari and Alanazi (2020) for an extensive study on tagging schemes in named entity recognition. In addition to the classes/labels, there usually also exists an *"other"* or *"outside"* class ("O").

2.2 Data Collection

To apply weak supervision unlabeled data needs to be collected, which can then be used to train a machine learning system. To form a well-suited training data set several design choices have to be specified in advance. This section discusses the requirements for the unlabeled data. Note that this data cannot be used for supervised learning without labels and is only one of the first steps on the way to the conversion to a weakly supervised machine learning system.

One of the most basic questions is determining the size of the data set. Usually acquiring labeled data comes with high costs of annotations. If weak supervision already resolves this cost problem, one is still confronted with the question of how many data samples should be labeled for training. In industry (as in most other contexts) data is not unlimited, thus a good starting point is just to use what is there. One way to estimate a suitable amount of data points is to produce a learning curve, that can display performance plateaus (Perlich, 2010). If only little data is available, up-sampling methods can be used to increase dataset size (e.g. (Feng et al., 2021)).

Since the trained model should be applicable in practice, the training data should reflect reality at inference time as closely as possible. Therefore, the real-world data should be examined beforehand. If there are large differences in the real-world data, these should also be represented by the training data set. In the case of the supplier mails the different suppliers contacts have a different writing style and some are not native speakers. While some include the relevant information into the body text, others represent them with bullet points or attachments. The data set should contain sufficient examples for all varieties, so that during inference all data peculiarities can be handled – otherwise there is an increased risk that the model suffers from biases and domain shift.

2.3 Labeling Function Identification

Weak supervision can be formalized as the labeling of large amounts of data by using labeling functions (LFs). Hence, we aim to retrieve useful LFs from the rule-based system.

LFs are programmatic rules to detect a prespecified configuration in a data point and each LF is associated with a specific class, thus it is possible to assign the respective label automatically. Formally, assume we have $|X| = n$ data points and $|Y| = m$ possible labels. To retrieve weakly supervised labels for X we define LFs $\lambda = (\lambda_1, ..., \lambda_l)$, $\lambda_i : X \to \emptyset \cup \{y\}$. Each LF either abstains from labeling ($\emptyset$) or assigns a class label y.

Labeling by LFs can be expressed using two matrices: i) a matrix of LF matches Z, where $Z \in \{0,1\}^{n \times l}$, with $Z_{i,j} = 1$ if LF j matches instance i, otherwise $Z_{i,j} = 0$. ii) a matrix encoding the link from LFs to the labels T, where $T \in \{0,1\}^{l \times m}$, $T_{j,k} = 1$ if label k is associated

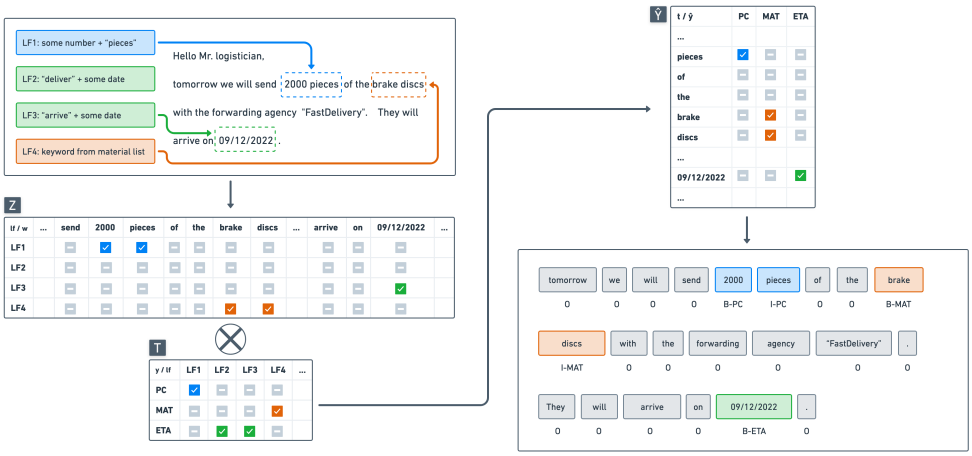


Figure 3: Stages of the labeling process. The LFs are applied to the input to acquire matrix Z. Matrix T contains the mapping of LFs λ to labels y. Combining both results in noisy labels for the input text and matrix $\hat{Y}$.

with LF j, $T_{j,k} = 0$ otherwise. See Figure 3 to see how the LFs are applied to the text and matrices Z and T are retrieved. Usually the information for T is gathered at the time the LFs are identified.

In rule-based systems there are databases, regular expressions, keywords, parsing outputs and other code fragments that extract the desired information and could potentially be used to define LFs. We consider these as sources of information and identifying them in heterogeneous code bases is a difficult task. Especially, when the person that tries to detect the LFs did not implement the rule-based system, or when there is a pipeline of different modules involved. The most important step is to identify the pieces of code where rule-based decisions are made. Sometimes it is difficult to determine which piece of code is ultimately responsible for a decision, or a decision is spread over several pieces of code. In this case, one can either define LFs for all of the pieces individually or merge the output of the pieces into one single LF.

In our running example, supplier emails, the existing rule-based system is modular. See Figure 4 for a simplified overview of the LF extraction process. The input mails are passed through the single modules with each module having a different functionality, e.g. markup removal and text cleaning, recognizing time expressions, checking material numbers. If one of the modules would output a label and we identified the source of information, a LF can be created from the corresponding piece of code in the module. For example for material mentions there is a *Material Crawler* (module) that searches every input text for material numbers from a delivery database (source of information). Thus, we can either specify a coarse LF *LF:database_match* or a more fine-grained one where each number forms a single LF like a keyword: *LF:numberXY_match*.

In general, there can be a degree of freedom in what level of granularity to reflect in the LFs. Extremes would be combining the LFs in such a way that they only express the final class label, or on the other hand using each LF individually. We recommend to reflect a fine-grained granularity in the LFs, that potentially allows denoising methods to later filter out specific LFs based on their reliability.

2.4 Weak Supervision

Now that we have the matrices Z and T, the next step is model training. Traditionally, many systems require labels that would have to be created from Z and T, arriving with the matrix $\hat{Y}$ containing noisy labels. Alternatively, there are end-to-end systems that can take matrices Z and T as input. Examples of both types are explained in section 3 in more detail.

In the following we describe two traditional approaches to create labels from Z and T. We aim to produce "weak" labels from the rule matches, that will be stored in matrix $\hat{Y}$, where $\hat{Y} \in \{0, ..., m\}^n$, with $\hat{Y}_i$ expressing the label index of instance i (in sequence labeling: position i). When generated,

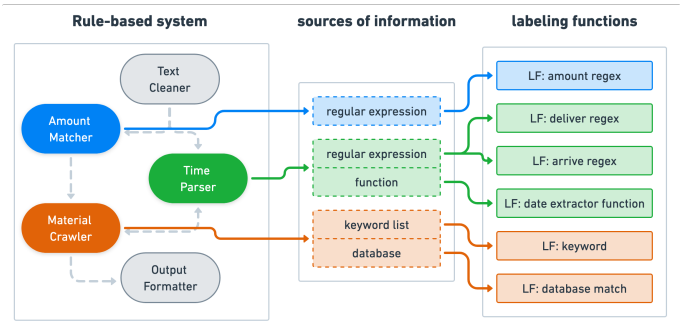


Figure 4: Overview of the process of retrieving LFs from the rule-based system. Modules output certain information pieces which are considered as sources of information. Finally, this information is encoded in LFs.

the labels $\hat{Y}$ can be used with any arbitrary standard supervised machine learning systems that takes labels for model training. It is not uncommon that several LFs match for one token. Such cases can be resolved by majority voting, where the majority of LF matches determines the label. If no clear majority exists either a label can be chosen randomly or the "other" class can be assigned.

Importantly, there still needs to be a manually labeled test set. We argue that annotation efforts should rather go into the labeling of evaluation than training data, since it is impossible to estimate the quality of the algorithm without reliable gold standard labels. In the following we point out a few important clues how to create a viable test set. For most industrial tasks, manual labeling has to be performed by domain experts rather than data scientists or developers. Therefore, annotation guidelines should be created to ensure that the labels assigned to the data are the same, regardless of the annotator. We recommend to choose a test data set that is sufficiently large to reliably indicate differences in model performance. In our experience, test data sets should be at least 100 samples (i.e. model decisions) large (for a very coarse detection of large model differences), but better in the order of 1000 samples. In our experiments, we use a test set of around 6K tokens.

3 Case Study: Weak Supervision in the Automotive Domain

This section guides through the process of turning a rule-based model into a machine learning system using the supplier email example. After the presentation of the problem, including data and annotation description, a detailed experimentation is shown.

3.1 Problem Description

We frame this information extraction problem as a sequence tagging task that addresses named entity recognition with custom entities.

The data for the experiments presented in this work stems from supplier emails in an automotive company. Since we aim to extract content from the email body, we remove header and footer in a preprocessing step. The remaining body text is mostly unstructured. The emails are sent by different suppliers and thus vary in writing style and presentation of the information. While some suppliers include all necessary information in the body text, others present the information in bullet points or tables. For our experiments, we do not take attachments or tables within the email into account and instead focus on the unstructured body text. One email can include information about materials, material numbers, the amount of the material and the estimated time of departure and/or arrival. As for standard named entity recognition we do not expect that every instance contains any or all entities from our entity set. The entities to be retrieved from the emails are the following: i) **MAT**: mentioned material number, ii) **PC**: how many pieces/ which amount of the material the information is about, iii) **ETD**: estimated time of departure of the delivery, iv) **ETA**: estimated time of arrival of the delivery, v) **TRG**: trigger token that indicates if a time is ETD or ETA.

Every token that is not assigned one of the above mentioned entities is annotated with an **O** (outside) tag. Following the usual convention of named entity tagging, we use the IOB2 tagging

scheme (Tjong Kim Sang and Veenstra, 1999) for our dataset.

3.2 Transformation to Weak Supervision

For the training data set we compose an email set from different postbox dumps of the logisticians. This set contains mails that are relevant for the task (supplier mails mentioning materials and deliveries) as well as emails that are irrelevant for our task (other conversations without delivery mentions). As described in section 2.3 a rule-based system is utilized to identify LFs so that it produces the matrix Z. More specifically, we identify the spots in the code that make a relevant decision (generate a label) and save their output. This file in turn was then processed by a custom parser so that the relevant information could also be assigned to the correct tokens in the instance. Since the existing code is highly encapsulated, this process required a good understanding of the entire rule-based system. We create the matrix T of size 5×9 (number of classes $\times$ number of LFs) semi-automatically using keywords. Via majority vote the weak labels are assigned for each token in the input emails, resulting in $\hat{Y}$. See Table 1 for an overview of corpus statistics.

Our gold standard test set contains 182 manually annotated emails, which comprise around 6K tokens. The annotations are based on annotation guidelines that were designed together with the domain experts beforehand.

3.3 Experimental Setup

The data is split into 80% for the test set and 20% for the validation set. We perform a hyperparameter grid search for our models and report hyperparameters of the final models as well as the grids in the appendix. If we use pre-trained language models (like BERT or RoBERTa) we freeze the encoder and only fine-tune the token classification head.

We experiment with two different types of approaches in this work. On the one hand we train taggers designed for standard sequence tagging: FLAIR and RoBERTa. In addition we experiment with three methods that are designed for weak supervision and also take the LFs/ rule matches into account: HMM, CHMM and SeqKnowMan.

We evaluate our results using the conlleval[1] script from the CoNLL-2000 shared task and report micro F1 scores.

[1] https://www.cnts.ua.ac.be/conll2000/chunking/output.html

split	# emails	# tokens
train	10133	978313
dev	3452	243830
test	182	5927

Table 1: Sizes of the data splits of the email corpus.

3.4 Models

Majority Vote Baseline The naive *MV baseline* is to apply the LFs to the test data directly and retrieve labels via majority vote. This corresponds to the rule-based system.

RoBERTa We perform standard supervised learning by training a pre-trained RoBERTa model (Liu et al., 2019) on the majority vote labels and refer to it as *RoBERTa MV*.

FLAIR Since FLAIR (Akbik et al., 2019) is one of the state-of-the-art tools for named entity recognition, we train different FLAIR models on the MV labels, using the following embedding stacks: FLAIR_*base* uses fastText[2] and character-level embeddings only, FLAIR_*bert* uses BERT embeddings on top, and FLAIR_*roberta* uses RoBERTa embeddings additionally.

SeqKnowMan Recently, KnowMan (März et al., 2021), an architecture for learning representations that are invariant to the noisy weakly supervised input signals, was developed. KnowMan originally was designed to tackle classification tasks. For the email data task, we adapt KnowMan and present SeqKnowMan that can also predict on the token level. Note, that SeqKnowMan is trained not only on the MV labels, but also requires matrix Z.

HMM and CHMM Traditionally, HMMs (Hidden Markov models) are used for temporal modeling. Lison et al. (2020) represent true labels as latent variables in an HMM and learn using expectation maximization. Additionally, (Li et al., 2021) proposed an approach for a conditional Hidden Markov model-based label model where the Markov model is enriched with contextual representations from BERT. We use the implementation of (Zhang et al., 2021) to train the HMM/CHMM label model and also use their LSTM and BERT-based end models for prediction.

3.5 Results

As the results in Table 2 show the *MV baseline* can achieve an F1 score of 62.9. *RoBERTa MV* per-

[2] https://fasttext.cc/

model	prec	rec	F1
MV baseline	75.8	51.6	62.9
RoBERTa MV	69.0	60.9	**64.7**
SeqKnowMAN l0	70.4	55.3	**61.9**
SeqKnowMAN l05	95.4	11.1	19.9
SeqKnowMAN l1	88.5	14.2	24.4
FLAIR_base	83.5	39.3	53.5
FLAIR_BERT	79.4	54.1	**64.4**
FLAIR_RoBERTa	83.8	44.3	58.0
HMM LSTM	70.3	32.0	44.0
HMM BERT	76.9	20.84	32.8
CHMM LSTM	81.4	29.8	43.63
CHMM BERT	69.73	22.1	33.6

Table 2: Overview of the results on the test set reported in **prec**ision, **rec**all and micro F1 scores.

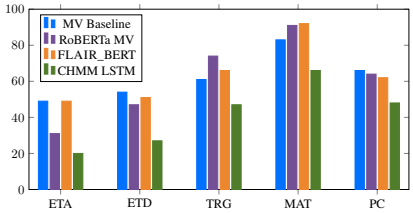


Figure 5: F1 scores for each tag across the best performing models.

forms best across all models and reaches a score of 64.7 F1. The performance of SEQKNOWMAN is with 61.9 F-score also similar to the baseline as long as no lambda value is set, i.e. as long as it is not attempted to blur the information contained in the LFs. The KNOWMAN models that are trained towards LF invariance yield very poor results, see SEQKNOWMAN *l05* and SEQKNOWMAN *l1* in Table 2. We hypothesize that each LF carries valuable information, so training a model which is invariant to a specific LF does not obtain sufficiently useful information.

For the FLAIR models only FLAIR_*BERT* performs close to *RoBERTa MV* with an F1 score of 64.4. Though, it is possible that the FLAIR taggers did not reach their full potential. A hyperparameter grid search showed that the results are very different depending on the configuration. This shows how challenging the custom named entity recognition task is on this data.

Using a label model to retrieve aggregated labels is harmful for our setup. The performance of the *HMM* and the *CHMM* models is noticeably lower than for the other taggers. This can be explained by the fact that the label models only produce accuracy values of around 75%, so it is likely that the end models are trained on some wrong labels.

We draw another finding from the precision and recall results for the models. The two models that outperform the *MV baseline*, as well as SEQKNOWMAN_*l0* achieve higher recall values. We conclude that weak supervision enables the retrieval of more tags compared to the rule-based system, thus generalizing from the labeling functions towards other important patterns. In contrast their precision values are not necessarily higher then for the *MV baseline*.

However, since the best model (*RoBERTa MV* can outperform the rule-based baseline we claim that one can actually train good models with no performance loss, while requiring less maintenance. Once trained, the models can be reused without great effort and thus make the maintenance of complex rule-based systems unnecessary.

3.6 Error Analysis

In order to get a deeper insight in the performance of the models, we take a close look at the data to identify instances that reflect the most common errors made by the models. See Figure 6 for three example sentences from the manually annotated test set (translated from German to English). There are many instances in the data where ETD and ETA are mentioned in one sentence. We observe that it is hard for the *MV Baseline* to label both mentions correctly at the same time. The rule-based model will only predict "ETA" as the label for both dates, regardless of whether the trigger comes before or after the time mentions.The third example sentence shows that if the sentence has no standard syntax, but is e.g., part of an enumeration, the *MV Baseline* is only able to label MAT correctly.

Another interesting observation can be drawn from the F1 score per label (see Figure 5). The MAT label is the easiest to predict for the models, since the material numbers all have a very similar surface forms. However, the taggers have the hardest time predicting the ETA/ETD labels. The time mentions in the text may be i) expressing ETD in a sentence, ii) expressing ETA or iii) expressing none of both, because the context is not about a delivery. Consequently, these labels are actually very difficult to predict correctly.

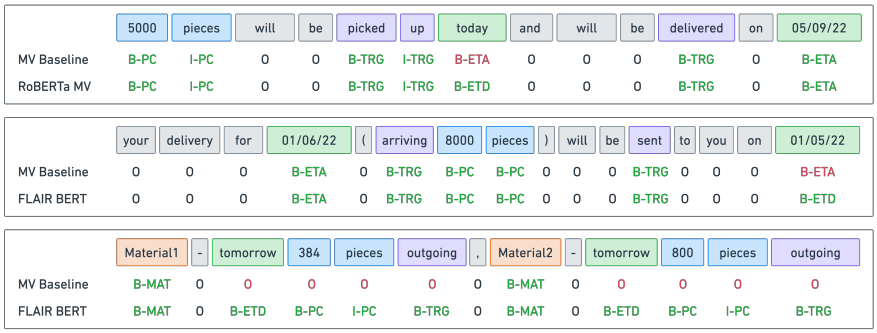


Figure 6: Example sentences translated from the test set. Correctly predicted tags are highlighted in green, wrongly predicted tags are indicated in red.

4 Related Work

Weak supervision has frequently been used in industry. The advantage of being able to generate large amounts of data quickly and relatively easily is compelling. Since one of the biggest challenges in this context is to create good and practical labeling functions (LFs), Cohen-Wang et al. (2019) and Evensen et al. (2020) propose methods how to easily retrieve and design LFs manually. However, the design and manual creation of LFs is not the focus of our investigations.

There are few published approaches on how to use weak supervision from existing systems. Similar to our work, Bach et al. (2018) also have recognized the need to apply weak supervision and conducted several experiments with industry data. The difference to our work is that there is no detailed description how *existing* systems can actually be converted into LFs. In addition, Bach et al. (2018) focuses on the Snorkel (Ratner et al., 2017) label model, and do not study alternative weak supervision architectures.

With the advent of the data programming paradigm (Ratner et al., 2016) and data centric AI (e.g. (Koch et al., 2021) or (Mazumder et al., 2022)), there were some approaches that tried to create a better data basis for their systems using weak supervision. Therefore, there are some approaches on how data programming and thus also LFs can be used well in the industrial environment. Ein-Dor et al. (2019) address financial event extraction and compare a weakly supervised and a manually labeled data set, similar to our test set. Like us, they use BERT as the end-model. In contrast to us, however, they do not use the weak labels themselves for further studies. Their focus is more on the data sources themselves and from which sources the LFs could be created. Like us, Sun et al. (2021) use data on a large scale in industry cases utilizing weak supervision. They also do use the Snorkel framework to retrieve labels for the data and conduct experiments with this weakly supervised data. Overall, they find that the usage of weak supervision works well, but there is no integration of existing systems to create LFs. Instead, LFs are created manually. Chatterjee et al. (2020) tackle scientific datasets and claim that label models are unstable and susceptible to different parameter combinations. They propose the integration of a quality guide and continuous LFs.

5 Conclusion

We present a case study on how to turn a rule-based industry information extraction system into a machine learning model by utilizing weak supervision. We explain how to retrieve labeling functions from a rule-based system and what requirements are necessary to produce weakly supervised data and train a machine learning model. We illustrate the process using an example from the automotive industry that tackles custom named entity recognition. The results of our experiments confirm that weak supervision for named entity recognition improves on the status quo. Thus, we conclude that rule-based systems can be reused to train flexible and powerful neural models, which can then in turn be used without the enormous maintenance effort of outdated systems. This work is aimed at practitioners who want to use machine learning models for their individual problems in a company. Our guide can be applied for numerous problems and end tasks.

Limitations

As part of our experiments, we found out that the sentence-wise processing likely misses some cases where the correct label can only be determined with certainty based on the preceding sentence. However, processing the entire email as input was not possible because the instances were simply too long to be processed with standard transformer language models. Thus the sentence-based sequence tagging might by a limitation for this setup. To solve this problem, one would have to examine approaches specifically designed for processing long documents.

Another point is that through the process described in this work, one can generate very large datasets to use them for machine learning. This in turn presupposes that suitable resources (e.g. memory space and GPUs) must also be available in order to effectively train this model. This can be a challenge (especially for small companies) that would have to be solved beforehand.

Additionally, in practice, it can be difficult to estimate the complexity of transforming a rule-based system as described, as there are many uncertainties. Among others, the code quality of the legacy system or organizational barriers such as code access might hinder the process. Furthermore the quantity, quality and coverage of the rules are difficult to estimate beforehand. Nevertheless, we believe that a successfully and carefully executed project, following the outlined steps, can successfully transform an existing rule-based system into a machine learning model.

References

Alan Akbik, Tanja Bergmann, Duncan Blythe, Kashif Rasul, Stefan Schweter, and Roland Vollgraf. 2019. FLAIR: An easy-to-use framework for state-of-the-art NLP. In *NAACL 2019, 2019 Annual Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 54–59.

Nasser Alshammari and Saad Alanazi. 2020. The impact of using different annotation schemes on named entity recognition. *Egyptian Informatics Journal*, 22.

Stephen H. Bach, Daniel Rodriguez, Yintao Liu, Chong Luo, Haidong Shao, Cassandra Xia, Souvik Sen, Alexander Ratner, Braden Hancock, Houman Alborzi, Rahul Kuchhal, Christopher Ré, and Rob Malkin. 2018. Snorkel drybell: A case study in deploying weak supervision at industrial scale. *CoRR*, abs/1812.00417.

Oishik Chatterjee, Ganesh Ramakrishnan, and Sunita Sarawagi. 2020. Robust data programming with precision-guided labeling functions. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(04):3397–3404.

Benjamin Cohen-Wang, Stephen Mussmann, Alex Ratner, and Chris Ré. 2019. Interactive programmatic labeling for weak supervision. In *Proc. KDD DCCL Workshop*, volume 120.

Liat Ein-Dor, Ariel Gera, Orith Toledo-Ronen, Alon Halfon, and Benjamin Sznajder. 2019. Financial event extraction using wikipedia-based weak supervision. *CoRR*, abs/1911.10783.

Sara Evensen, Chang Ge, Dongjin Choi, and Çagatay Demiralp. 2020. Data programming by demonstration: A framework for interactively learning labeling functions. *CoRR*, abs/2009.01444.

Steven Y. Feng, Varun Gangal, Jason Wei, Sarath Chandar, Soroush Vosoughi, Teruko Mitamura, and Eduard H. Hovy. 2021. A survey of data augmentation approaches for NLP. *CoRR*, abs/2105.03075.

Bernard Koch, Emily Denton, Alex Hanna, and Jacob G. Foster. 2021. Reduced, reused and recycled: The life of a dataset in machine learning research. *CoRR*, abs/2112.01716.

Yinghao Li, Pranav Shetty, Lucas Liu, Chao Zhang, and Le Song. 2021. Bertifying the hidden markov model for multi-source weakly supervised named entity recognition. *CoRR*, abs/2105.12848.

Pierre Lison, Jeremy Barnes, Aliaksandr Hubin, and Samia Touileb. 2020. Named entity recognition without labelled data: A weak supervision approach. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1518–1533, Online. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

Luisa März, Ehsaneddin Asgari, Fabienne Braune, Franziska Zimmermann, and Benjamin Roth. 2021. Knowman: Weakly supervised multinomial adversarial networks. *CoRR*, abs/2109.07994.

Mark Mazumder, Colby Banbury, Xiaozhe Yao, Bojan Karlaš, William Gaviria Rojas, Sudnya Diamos, Greg Diamos, Lynn He, Douwe Kiela, David Jurado, David Kanter, Rafael Mosquera, Juan Ciro, Lora Aroyo, Bilge Acun, Sabri Eyuboglu, Amirata Ghorbani, Emmett Goodman, Tariq Kane, Christine R. Kirkpatrick, Tzu-Sheng Kuo, Jonas Mueller, Tristan Thrush, Joaquin Vanschoren, Margaret Warren, Adina Williams, Serena Yeung, Newsha Ardalani, Praveen Paritosh, Ce Zhang, James Zou, Carole-Jean Wu, Cody Coleman, Andrew Ng, Peter Mattson, and

Vijay Janapa Reddi. 2022. Dataperf: Benchmarks for data-centric ai development.

Claudia Perlich. 2010. Learning curves in machine learning.

Alexander Ratner, Stephen H. Bach, Henry R. Ehrenberg, Jason Alan Fries, Sen Wu, and Christopher Ré. 2017. Snorkel: Rapid training data creation with weak supervision. *CoRR*, abs/1711.10160.

Alexander J Ratner, Christopher M De Sa, Sen Wu, Daniel Selsam, and Christopher Ré. 2016. Data programming: Creating large training sets, quickly. In *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc.

Yixuan Sun, Tanwi Mallick, Prasanna Balaprakash, and Jane MacFarlane. 2021. A data-centric weak supervised learning for highway traffic incident detection. *CoRR*, abs/2112.09792.

Erik F. Tjong Kim Sang and Jorn Veenstra. 1999. Representing text chunks. In *Ninth Conference of the European Chapter of the Association for Computational Linguistics*, pages 173–179, Bergen, Norway. Association for Computational Linguistics.

Jieyu Zhang, Yue Yu, Yinghao Li, Yujing Wang, Yaming Yang, Mao Yang, and Alexander Ratner. 2021. WRENCH: A comprehensive benchmark for weak supervision. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

A Model Training Details

We ran all our experiments on a DGX-1 server with one V100 GPU per experiment. See Table 3 for run times and hyperparameters of the different models, if not listed explicitly we use the default configurations of the models. FLAIR models are trained using the standard FLAIR library, HMM and CHMM are trained using the WRENCH implementation[3]. RoBERTa MV and SeqKnowMAN are trained with our own implementations.

B Results on the dev set

Table 4 shows the F1 scores for the different models on the development set. Note, that the development set is part of our weakly supervised data and not a manual labeled gold-standard as the test set. Thus model performance differs from the results on the test set.

[3] https://github.com/JieyuZ2/wrench

model	runtime	batch size	learning rate	epochs/ steps
MV baseline	10 h	-	-	-
RoBERTa MV	2 h	[8, **16**, 32]	[0.1, 0.01, **0.001**, 0.0001]	[2, **5**, 10]
SeqKnowMAN l0	2 h	[8, **16**, 32]	[0.1, 0.01, **0.001**, 0.0001]	[2, **5**, 10]
SeqKnowMAN l05	2 h	[8, **16**, 32]	[0.1, 0.01, **0.001**, 0.0001]	[2, **5**, 10]
SeqKnowMAN l1	2 h	[8, **16**, 32]	[0.1, 0.01, **0.001**, 0.0001]	[2, **5**, 10]
FLAIR_base	10 h	[**8**, 16, 32]	[**0.1**, 0.01, 0.001, 0.0001]	[2, 5 **15**]
FLAIR_BERT	22 h	[**8**, 16, 32]	[0.1, **0.01**, 0.001, 0.0001]	[2, 5 **15**]
FLAIR_RoBERTa	22 h	[**8**, 16, 32]	[0.1, **0.01**, 0.001, 0.0001]	[2, 5 **15**]
HMM	0.1 h	-	-	[**15**, 25, 50]
CHMM	1.5 h	[8, 16, **32**]	-	[**15**, 25, 50]
LSTM	1 h	[8, 16, **32**]	[0.1, 0.01, **0.001**, 0.0001]	10K
BERT	1 h	[**8**, 16, 32]	[0.00001, **0.00002**]	2K

Table 3: Overview of run times in hours per parameter configuration and hyperparameters per model. We report the number of epochs for SeqKnowMAN and FLAIR models. For LSTM/BERT models we report the number of steps. Bold numbers indicate the chosen hyperparameters for the final models.

model	dev F1
RoBERTa MV	48.8
SeqKnowMAN l0	48.5
SeqKnowMAN l05	57.6
SeqKnowMAN l1	49.5
FLAIR_base	70.3
FLAIR_BERT	70.7
FLAIR_RoBERTa	71.7
HMM LSTM	62.3
HMM BERT	55.7
CHMM LSTM	64.5
CHMM BERT	59.0

Table 4: Overveiw of the results on the dev set reported in micro F1 scores.

Bibliography

Alexandra Y Aikhenvald et al. Grammars in contact: A cross-linguistic perspective. *Grammars in contact: A cross-linguistic typology*, 4:1, 2006.

Alan Akbik, Duncan Blythe, and Roland Vollgraf. Contextual string embeddings for sequence labeling. In Emily M. Bender, Leon Derczynski, and Pierre Isabelle, editors, *Proceedings of the 27th International Conference on Computational Linguistics, COLING 2018, Santa Fe, New Mexico, USA, August 20-26, 2018*, pages 1638–1649. Association for Computational Linguistics, 2018. URL `https://aclanthology.org/C18-1139/`.

Liaqat Ali, Awais Niamat, Javed Ali Khan, Noorbakhsh Amiri Golilarz, Xiong Xingzhong, Adeeb Noor, Redhwan Nour, and Syed Ahmad Chan Bukhari. An optimized stacked support vector machines based expert system for the effective prediction of heart failure. *IEEE Access*, 7:54007–54014, 2019a.

Najat Ali, Daniel Neagu, and Paul R. Trundle. Evaluation of k-nearest neighbour classifier performance for heterogeneous data sets. *SN Applied Sciences*, 2019b.

Nasser O. Alshammari and Saad Awadh Alanazi. The impact of using different annotation schemes on named entity recognition. *Egyptian Informatics Journal*, 2020.

Stephen H. Bach, Bryan Dawei He, Alexander Ratner, and Christopher Ré. Learning the structure of generative models without labeled data. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 273–282. PMLR, 2017. URL `http://proceedings.mlr.press/v70/bach17a.html`.

Stephen H. Bach, Daniel Rodriguez, Yintao Liu, Chong Luo, Haidong Shao, Cassandra Xia, Souvik Sen, Alexander Ratner, Braden Hancock, Houman Alborzi, Rahul Kuchhal, Christopher Ré, and Rob Malkin. Snorkel drybell: A case study in deploying weak supervision at industrial scale. In Peter A. Boncz, Stefan Manegold, Anastasia Ailamaki, Amol Deshpande, and Tim Kraska, editors, *Proceedings of the 2019 International Conference on Management of Data, SIGMOD Conference 2019, Amsterdam, The Netherlands, June 30 - July 5, 2019*, pages 362–375. ACM, 2019. doi: 10.1145/3299869.3314036. URL `https://doi.org/10.1145/3299869.3314036`.

Hyeong-Ryeol Baek and Yong-Suk Choi. Enhancing targeted minority class prediction in sentence-level relation extraction. *Sensors*, 22(13):4911, 2022. doi: 10.3390/s22134911. URL `https://doi.org/10.3390/s22134911`.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1409.0473.

Benedikt Boecking, Willie Neiswanger, Eric P. Xing, and Artur Dubrawski. Interactive weak supervision: Learning useful heuristics for data labeling. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=IDFQI9OY6K.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomás Mikolov. Enriching word vectors with subword information. *Trans. Assoc. Comput. Linguistics*, 5:135–146, 2017. doi: 10.1162/tacl_a_00051. URL https://doi.org/10.1162/tacl_a_00051.

Bernhard E. Boser, Isabelle Guyon, and Vladimir Vapnik. A training algorithm for optimal margin classifiers. In David Haussler, editor, *Proceedings of the Fifth Annual ACM Conference on Computational Learning Theory, COLT 1992, Pittsburgh, PA, USA, July 27-29, 1992*, pages 144–152. ACM, 1992. doi: 10.1145/130385.130401. URL https://doi.org/10.1145/130385.130401.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html.

Salva Rühling Cachay, Benedikt Boecking, and Artur Dubrawski. End-to-end weak supervision. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 1845–1857, 2021a. URL https://proceedings.neurips.cc/paper/2021/hash/0e674a918ebca3f78bfe02e2f387689d-Abstract.html.

Salva Rühling Cachay, Benedikt Boecking, and Artur Dubrawski. Dependency structure misspecification in multi-source weak supervision models. *CoRR*, abs/2106.10302, 2021b. URL https://arxiv.org/abs/2106.10302.

Rich Caruana and Alexandru Niculescu-Mizil. An empirical comparison of supervised learning algorithms. In William W. Cohen and Andrew W. Moore, editors, *Machine Learning, Proceedings of the Twenty-Third International Conference (ICML 2006), Pittsburgh, Pennsylvania, USA, June 25-29, 2006*, volume 148 of *ACM International Conference Proceeding Series*, pages 161–168. ACM, 2006. doi: 10.1145/1143844.1143865. URL https://doi.org/10.1145/1143844.1143865.

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Brian Strope, and Ray Kurzweil. Universal sentence encoder for english. In Eduardo Blanco and Wei Lu, editors, *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, EMNLP 2018: System Demonstrations, Brussels, Belgium, October 31 - November 4, 2018*, pages 169–174. Association for Computational Linguistics, 2018. doi: 10.18653/v1/d18-2029. URL https://doi.org/10.18653/v1/d18-2029.

Branden Chan, Stefan Schweter, and Timo Möller. German's next language model. In Donia Scott, Núria Bel, and Chengqing Zong, editors, *Proceedings of the 28th International Conference on Computational Linguistics, COLING 2020, Barcelona, Spain (Online), December 8-13, 2020*, pages 6788–6796. International Committee on Computational Linguistics, 2020. doi: 10.18653/v1/2020.coling-main.598. URL https://doi.org/10.18653/v1/2020.coling-main.598.

Sneha Chaudhari, Varun Mithal, Gungor Polatkan, and Rohan Ramanath. An attentive survey of attention models. *ACM Trans. Intell. Syst. Technol.*, 12(5):53:1–53:32, 2021. doi: 10.1145/3465055. URL https://doi.org/10.1145/3465055.

Stanley F Chen. *Building probabilistic models for natural language*. Harvard University, 1996.

Nancy Chinchor. Appendix E: MUC-7 named entity task definition (version 3.5). In *Seventh Message Understanding Conference: Proceedings of a Conference Held in Fairfax, Virginia, USA, MUC 1998, April 29 - May 1, 1998*. ACL, 1998. URL https://aclanthology.org/M98-1028/.

Noam Chomsky. Three models for the description of language. *IRE Trans. Inf. Theory*, 2 (3):113–124, 1956. doi: 10.1109/TIT.1956.1056813. URL https://doi.org/10.1109/TIT.1956.1056813.

Amir D. N. Cohen, Shachar Rosenman, and Yoav Goldberg. Relation extraction as two-way span-prediction. *CoRR*, abs/2010.04829, 2020. URL https://arxiv.org/abs/2010.04829.

Ronan Collobert and Jason Weston. A unified architecture for natural language processing: deep neural networks with multitask learning. In *ICML '08*, 2008.

Thomas M. Cover and Peter E. Hart. Nearest neighbor pattern classification. *IEEE Trans. Inf. Theory*, 13(1):21–27, 1967. doi: 10.1109/TIT.1967.1053964. URL https://doi.org/10.1109/TIT.1967.1053964.

Haskell B. Curry. The method of steepest descent for non-linear minimization problems. *Quarterly of Applied Mathematics*, 2:258–261, 1944.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics, 2019. doi: 10.18653/v1/n19-1423. URL https://doi.org/10.18653/v1/n19-1423.

Siyu Ding, Junyuan Shang, Shuohuan Wang, Yu Sun, Hao Tian, Hua Wu, and Haifeng Wang. Ernie-doc: A retrospective long-document modeling transformer. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli, editors, *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pages 2914–2927. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.acl-long.227. URL https://doi.org/10.18653/v1/2021.acl-long.227.

Jiahua Dong, Yang Cong, Gan Sun, Zhen Fang, and Zhengming Ding. Where and how to transfer: Knowledge aggregation-induced transferability perception for unsupervised domain adaptation. *IEEE transactions on pattern analysis and machine intelligence*, PP, 2021.

Hao Fei, Yafeng Ren, Yue Zhang, Donghong Ji, and Xiaohui Liang. Enriching contextualized language model from knowledge graph for biomedical information extraction. *Briefings Bioinform.*, 22(3), 2021. doi: 10.1093/bib/bbaa110. URL https://doi.org/10.1093/bib/bbaa110.

Nicholas Frosst and Geoffrey E. Hinton. Distilling a neural network into a soft decision tree. In Tarek R. Besold and Oliver Kutz, editors, *Proceedings of the First International Workshop on Comprehensibility and Explanation in AI and ML 2017 co-located with 16th International Conference of the Italian Association for Artificial Intelligence (AI*IA 2017), Bari, Italy, November 16th and 17th, 2017*, volume 2071 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2017. URL http://ceur-ws.org/Vol-2071/CExAIIA_2017_paper_3.pdf.

Kunihiko Fukushima and Sei Miyake. Neocognitron: A new algorithm for pattern recognition tolerant of deformations and shifts in position. *Pattern Recognit.*, 15(6): 455–469, 1982. doi: 10.1016/0031-3203(82)90024-3. URL https://doi.org/10.1016/0031-3203(82)90024-3.

Philip Gage. A new algorithm for data compression. *The C Users Journal archive*, 12: 23–38, 1994.

Antonio Javier Gallego, Jorge Calvo-Zaragoza, Jose J. Valero-Mas, and Juan Ramón Rico-Juan. Clustering-based k-nearest neighbor classification for large-scale data with neural codes representation. *Pattern Recognit.*, 74:531–543, 2018.

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. Generative adversarial networks. *CoRR*, abs/1406.2661, 2014. URL `http://arxiv.org/abs/1406.2661`.

Ahmed Hamdi, Elvys Linhares Pontes, Emanuela Boros, Thi Tuyet Hai Nguyen, Günter Hackl, José G. Moreno, and Antoine Doucet. A multilingual dataset for named entity recognition, entity linking and stance detection in historical newspapers. In Fernando Diaz, Chirag Shah, Torsten Suel, Pablo Castells, Rosie Jones, and Tetsuya Sakai, editors, *SIGIR '21: The 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, Virtual Event, Canada, July 11-15, 2021*, pages 2328–2334. ACM, 2021. doi: 10.1145/3404835.3463255. URL `https://doi.org/10.1145/3404835.3463255`.

Benjamin Heinzerling and Michael Strube. Bpemb: Tokenization-free pre-trained subword embeddings in 275 languages. In Nicoletta Calzolari, Khalid Choukri, Christopher Cieri, Thierry Declerck, Sara Goggi, Kôiti Hasida, Hitoshi Isahara, Bente Maegaard, Joseph Mariani, Hélène Mazo, Asunción Moreno, Jan Odijk, Stelios Piperidis, and Takenobu Tokunaga, editors, *Proceedings of the Eleventh International Conference on Language Resources and Evaluation, LREC 2018, Miyazaki, Japan, May 7-12, 2018*. European Language Resources Association (ELRA), 2018. URL `http://www.lrec-conf.org/proceedings/lrec2018/summaries/1049.html`.

Iris Hendrickx, Su Nam Kim, Zornitsa Kozareva, Preslav Nakov, Diarmuid Ó Séaghdha, Sebastian Padó, Marco Pennacchiotti, Lorenza Romano, and Stan Szpakowicz. Semeval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals. In Eneko Agirre, Lluís Màrquez, and Richard Wicentowski, editors, *Proceedings of the Workshop on Semantic Evaluations: Recent Achievements and Future Directions, SEW@NAACL-HLT 2009, Boulder, CO, USA, June 4, 2009*, pages 94–99. Association for Computational Linguistics, 2009. URL `https://aclanthology.org/W09-2415/`.

Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. *CoRR*, abs/1503.02531, 2015. URL `http://arxiv.org/abs/1503.02531`.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9 (8):1735–1780, 1997. doi: 10.1162/neco.1997.9.8.1735. URL `https://doi.org/10.1162/neco.1997.9.8.1735`.

Lars Hoffmann, Christian Bartelt, and Heiner Stuckenschmidt. Knowledge injection via ml-based initialization of neural networks. In *CIKM Workshops*, 2021.

Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S. Weld. Knowledge-based weak supervision for information extraction of overlapping relations.

In Dekang Lin, Yuji Matsumoto, and Rada Mihalcea, editors, *The 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Proceedings of the Conference, 19-24 June, 2011, Portland, Oregon, USA*, pages 541–550. The Association for Computer Linguistics, 2011. URL https://aclanthology.org/P11-1055/.

S. Jayachitra and A. Prasanth. Multi-feature analysis for automated brain stroke classification using weighted gaussian naïve bayes classifier. *J. Circuits Syst. Comput.*, 30: 2150178:1–2150178:22, 2021.

Shaoxiong Ji, Shirui Pan, E. Cambria, Pekka Marttinen, and Philip S. Yu. A survey on knowledge graphs: Representation, acquisition, and applications. *IEEE Transactions on Neural Networks and Learning Systems*, 33:494–514, 2022.

Haiyun Jiang, Qiaoben Bao, Qiao Cheng, Deqing Yang, Li Wang, and Yanghua Xiao. Complex relation extraction: Challenges and opportunities. *CoRR*, abs/2012.04821, 2020. URL https://arxiv.org/abs/2012.04821.

Prashant Johri, Sunil K Khatri, Ahmad T Al-Taani, Munish Sabharwal, Shakhzod Suvanov, and Avneesh Kumar. Natural language processing: History, evolution, application, and future work. In *Proceedings of 3rd International Conference on Computing Informatics and Networks*, pages 365–375. Springer, 2021.

Dan Jurafsky and James H. Martin. Speech and language processing - an introduction to natural language processing, computational linguistics, and speech recognition. In *Prentice Hall series in artificial intelligence*, 2000.

Sotiris B. Kotsiantis. Supervised machine learning: A review of classification techniques. In Ilias Maglogiannis, Kostas Karpouzis, Manolis Wallace, and John Soldatos, editors, *Emerging Artificial Intelligence Applications in Computer Engineering - Real Word AI Systems with Applications in eHealth, HCI, Information Retrieval and Pervasive Technologies*, volume 160 of *Frontiers in Artificial Intelligence and Applications*, pages 3–24. IOS Press, 2007. URL http://www.booksonline.iospress.nl/Content/View.aspx?piid=6950.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. Neural architectures for named entity recognition. In Kevin Knight, Ani Nenkova, and Owen Rambow, editors, *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, pages 260–270. The Association for Computational Linguistics, 2016. doi: 10.18653/v1/n16-1030. URL https://doi.org/10.18653/v1/n16-1030.

David D. Lewis and William A. Gale. A sequential algorithm for training text classifiers. In *SIGIR '94*, 1994.

Xiaoya Li, Xiaofei Sun, Yuxian Meng, Junjun Liang, Fei Wu, and Jiwei Li. Dice loss for data-imbalanced NLP tasks. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel R. Tetreault, editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 465–476. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.acl-main.45. URL https://doi.org/10.18653/v1/2020.acl-main.45.

Xin Li and Dan Roth. Learning question classifiers. In *19th International Conference on Computational Linguistics, COLING 2002, Howard International House and Academia Sinica, Taipei, Taiwan, August 24 - September 1, 2002*, 2002. URL https://aclanthology.org/C02-1150/.

Chen Liang, Yue Yu, Haoming Jiang, Siawpeng Er, Ruijia Wang, Tuo Zhao, and Chao Zhang. BOND: bert-assisted open-domain named entity recognition with distant supervision. In Rajesh Gupta, Yan Liu, Jiliang Tang, and B. Aditya Prakash, editors, *KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, CA, USA, August 23-27, 2020*, pages 1054–1064. ACM, 2020a. doi: 10.1145/3394486.3403149. URL https://doi.org/10.1145/3394486.3403149.

Jian Liang, Dapeng Hu, and Jiashi Feng. Do we really need to access the source data? source hypothesis transfer for unsupervised domain adaptation. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 6028–6039. PMLR, 2020b. URL http://proceedings.mlr.press/v119/liang20a.html.

Pierre Lison, Jeremy Barnes, and Aliaksandr Hubin. skweak: Weak supervision made easy for NLP. In Heng Ji, Jong C. Park, and Rui Xia, editors, *Proceedings of the Joint Conference of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL 2021 - System Demonstrations, Online, August 1-6, 2021*, pages 337–346. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.acl-demo.40. URL https://doi.org/10.18653/v1/2021.acl-demo.40.

Bin Liu, Chen-Chen Li, and Ke Yan. Deepsvm-fold: protein fold recognition by combining support vector machines and pairwise sequence similarity scores generated by deep learning networks. *Briefings in bioinformatics*, 2020.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692, 2019. URL http://arxiv.org/abs/1907.11692.

Yukun Ma, Haiyun Peng, and E. Cambria. Targeted aspect-based sentiment analysis via embedding commonsense knowledge into an attentive lstm. In *AAAI*, 2018.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In Dekang Lin, Yuji

Matsumoto, and Rada Mihalcea, editors, *The 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Proceedings of the Conference, 19-24 June, 2011, Portland, Oregon, USA*, pages 142–150. The Association for Computer Linguistics, 2011. URL https://aclanthology.org/P11-1015/.

Christopher Manning and Hinrich Schutze. *Foundations of statistical natural language processing*. MIT press, 1999.

Louis Martin, Benjamin Müller, Pedro Javier Ortiz Suárez, Yoann Dupont, Laurent Romary, Éric de la Clergerie, Djamé Seddah, and Benoît Sagot. Camembert: a tasty french language model. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel R. Tetreault, editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 7203–7219. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.acl-main.645. URL https://doi.org/10.18653/v1/2020.acl-main.645.

Luisa März, Dietrich Trautmann, and Benjamin Roth. Domain adaptation for part-of-speech tagging of noisy user-generated text. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 3415–3420. Association for Computational Linguistics, 2019. doi: 10.18653/v1/n19-1345. URL https://doi.org/10.18653/v1/n19-1345.

Yu Meng, Jiaming Shen, Chao Zhang, and Jiawei Han. Weakly-supervised neural text classification. In Alfredo Cuzzocrea, James Allan, Norman W. Paton, Divesh Srivastava, Rakesh Agrawal, Andrei Z. Broder, Mohammed J. Zaki, K. Selçuk Candan, Alexandros Labrinidis, Assaf Schuster, and Haixun Wang, editors, *Proceedings of the 27th ACM International Conference on Information and Knowledge Management, CIKM 2018, Torino, Italy, October 22-26, 2018*, pages 983–992. ACM, 2018. doi: 10.1145/3269206.3271737. URL https://doi.org/10.1145/3269206.3271737.

Tomás Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. In Christopher J. C. Burges, Léon Bottou, Zoubin Ghahramani, and Kilian Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*, pages 3111–3119, 2013. URL https://proceedings.neurips.cc/paper/2013/hash/9aa42b31882ec039965f3c4923ce901b-Abstract.html.

Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. Advances in pre-training distributed word representations. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*, 2018.

Fausto Milletari, Alex Rothberg, Jimmy Jia, and Michal Sofka. Integrating statistical prior knowledge into convolutional neural networks. In *MICCAI*, 2017.

Mike Mintz, Steven Bills, Rion Snow, and Daniel Jurafsky. Distant supervision for relation extraction without labeled data. In Keh-Yih Su, Jian Su, and Janyce Wiebe, editors, *ACL 2009, Proceedings of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the AFNLP, 2-7 August 2009, Singapore*, pages 1003–1011. The Association for Computer Linguistics, 2009. URL https://aclanthology.org/P09-1113/.

Seyed Iman Mirzadeh, Mehrdad Farajtabar, Ang Li, Nir Levine, Akihiro Matsukawa, and Hassan Ghasemzadeh. Improved knowledge distillation via teacher assistant. In *AAAI*, 2020.

Vladimir Nasteski. An overview of the supervised machine learning methods. *Horizons. b*, 4:51–62, 2017.

Deepak Nathani, Jatin Chauhan, Charu Sharma, and Manohar Kaul. Learning attention-based embeddings for relation prediction in knowledge graphs. In Anna Korhonen, David R. Traum, and Lluís Màrquez, editors, *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 4710–4723. Association for Computational Linguistics, 2019. doi: 10.18653/v1/p19-1466. URL https://doi.org/10.18653/v1/p19-1466.

An Thanh Nguyen, Byron C. Wallace, Junyi Jessy Li, Ani Nenkova, and Matthew Lease. Aggregating and predicting sequence labels from crowd annotations. In Regina Barzilay and Min-Yen Kan, editors, *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 299–309. Association for Computational Linguistics, 2017. doi: 10.18653/v1/P17-1028. URL https://doi.org/10.18653/v1/P17-1028.

Maximilian Nickel, Kevin P. Murphy, Volker Tresp, and Evgeniy Gabrilovich. A review of relational machine learning for knowledge graphs. *Proceedings of the IEEE*, 104:11–33, 2016.

Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Trans. Knowl. Data Eng.*, 22(10):1345–1359, 2010. doi: 10.1109/TKDE.2009.191. URL https://doi.org/10.1109/TKDE.2009.191.

Sinno Jialin Pan, Ivor W. Tsang, James T. Kwok, and Qiang Yang. Domain adaptation via transfer component analysis. *IEEE Trans. Neural Networks*, 22(2):199–210, 2011. doi: 10.1109/TNN.2010.2091281. URL https://doi.org/10.1109/TNN.2010.2091281.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In Alessandro Moschitti, Bo Pang, and Walter Daelemans, editors, *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1532–1543. ACL, 2014. doi: 10.3115/v1/d14-1162. URL https://doi.org/10.3115/v1/d14-1162.

J. Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993. ISBN 1-55860-238-0.

Dina Ahmed Ragab, Maha A. Sharkas, Stephen Marshall, and Jinchang Ren. Breast cancer detection using deep convolutional neural networks and support vector machines. *PeerJ*, 7, 2019.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100, 000+ questions for machine comprehension of text. In Jian Su, Xavier Carreras, and Kevin Duh, editors, *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 2383–2392. The Association for Computational Linguistics, 2016. doi: 10.18653/v1/d16-1264. URL https://doi.org/10.18653/v1/d16-1264.

Pierluigi Zama Ramirez, Alessio Tonioni, Samuele Salti, and Luigi Di Stefano. Learning across tasks and domains. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*, pages 8109–8118. IEEE, 2019. doi: 10.1109/ICCV.2019.00820. URL https://doi.org/10.1109/ICCV.2019.00820.

Alexander Ratner, Stephen H. Bach, Henry R. Ehrenberg, Jason Alan Fries, Sen Wu, and Christopher Ré. Snorkel: Rapid training data creation with weak supervision. *Proc. VLDB Endow.*, 11(3):269–282, 2017. doi: 10.14778/3157794.3157797. URL http://www.vldb.org/pvldb/vol11/p269-ratner.pdf.

Alexander J. Ratner, Christopher De Sa, Sen Wu, Daniel Selsam, and Christopher Ré. Data programming: Creating large training sets, quickly. In Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 3567–3575, 2016. URL https://proceedings.neurips.cc/paper/2016/hash/6709e8d64a5f47269ed5cea9f625f7ab-Abstract.html.

Pengzhen Ren, Yun Xiao, Xiaojun Chang, Po-Yao Huang, Zhihui Li, Brij B. Gupta, Xiaojiang Chen, and Xin Wang. A survey of deep active learning. *ACM Comput. Surv.*, 54(9):180:1–180:40, 2022. doi: 10.1145/3472291. URL https://doi.org/10.1145/3472291.

Wendi Ren, Yinghao Li, Hanting Su, David Kartchner, Cassie S. Mitchell, and Chao Zhang. Denoising multi-source weak supervision for neural text classification. In Trevor Cohn, Yulan He, and Yang Liu, editors, *Findings of the Association for Computational Linguistics: EMNLP 2020, Online Event, 16-20 November 2020*, volume EMNLP 2020 of *Findings of ACL*, pages 3739–3754. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.findings-emnlp.334. URL https://doi.org/10.18653/v1/2020.findings-emnlp.334.

Adam Roberts, Colin Raffel, and Noam Shazeer. How much knowledge can you pack into the parameters of a language model? In Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu, editors, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 5418–5426. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.emnlp-main.437. URL https://doi.org/10.18653/v1/2020.emnlp-main.437.

Frank Rosenblatt. Principles of neurodynamics. perceptrons and the theory of brain mechanisms. Technical report, Cornell Aeronautical Lab Inc Buffalo NY, 1961.

Benjamin Roth and Dietrich Klakow. Combining generative and discriminative model scores for distant supervision. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 24–29. ACL, 2013. URL https://aclanthology.org/D13-1003/.

David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning internal representations by error propagation. 1986.

Erik F. Tjong Kim Sang and Fien De Meulder. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In Walter Daelemans and Miles Osborne, editors, *Proceedings of the Seventh Conference on Natural Language Learning, CoNLL 2003, Held in cooperation with HLT-NAACL 2003, Edmonton, Canada, May 31 - June 1, 2003*, pages 142–147. ACL, 2003. URL https://aclanthology.org/W03-0419/.

Erik Tjong Kim Sang and Jorn Veenstra. Representing text chunks. In *EACL*, 1999.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of BERT: smaller, faster, cheaper and lighter. *CoRR*, abs/1910.01108, 2019. URL http://arxiv.org/abs/1910.01108.

Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117, 2015. doi: 10.1016/j.neunet.2014.09.003. URL https://doi.org/10.1016/j.neunet.2014.09.003.

Stefan Schweter and Alan Akbik. FLERT: document-level features for named entity recognition. *CoRR*, abs/2011.06993, 2020. URL https://arxiv.org/abs/2011.06993.

Joan Serrà, Didac Suris, Marius Miron, and Alexandros Karatzoglou. Overcoming catastrophic forgetting with hard attention to the task. In Jennifer G. Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 4555–4564. PMLR, 2018. URL http://proceedings.mlr.press/v80/serra18a.html.

Burr Settles. Active learning literature survey. 2009.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1631–1642. ACL, 2013. URL https://aclanthology.org/D13-1170/.

Hwanjun Song, Minseok Kim, Dongmin Park, and Jae-Gil Lee. Learning from noisy labels with deep neural networks: A survey. *CoRR*, abs/2007.08199, 2020. URL https://arxiv.org/abs/2007.08199.

Christos Troussas, Maria Virvou, Kurt Junshean Espinosa, Kevin Llaguno, and Jaime D. L. Caro. Sentiment analysis of facebook statuses using naive bayes classifier for language learning. *IISA 2013*, pages 1–6, 2013.

Alan M. Turing. Computing machinery and intelligence. *Mind*, LIX(236):433–460, 1950. doi: 10.1093/mind/LIX.236.433. URL https://doi.org/10.1093/mind/LIX.236.433.

Paroma Varma, Bryan D. He, Payal Bajaj, Nishith Khandwala, Imon Banerjee, Daniel L. Rubin, and Christopher Ré. Inferring generative model structure with static analysis. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 240–250, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/cedebb6e872f539bef8c3f919874e9d7-Abstract.html.

Laura von Rueden, Sebastian Mayer, Katharina Beckh, Bogdan Georgiev, Sven Giesselbach, Raoul Heese, Birgit Kirsch, Julius Pfrommer, Annika Pick, Rajkumar Ramamurthy, Michal Walczak, Jochen Garcke, Christian Bauckhage, and Jannis Schuecker. Informed machine learning - a taxonomy and survey of integrating prior knowledge into learning systems. *IEEE Transactions on Knowledge and Data Engineering*, 2021.

Chenguang Wang, Xiao Liu, Zui Chen, Haoyun Hong, Jie Tang, and Dawn Song. Deepstruct: Pretraining of language models for structure prediction. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio, editors, *Findings of the Association for Computational Linguistics: ACL 2022, Dublin, Ireland, May 22-27, 2022*, pages 803–823. Association for Computational Linguistics, 2022. doi: 10.18653/v1/2022.findings-acl.67. URL https://doi.org/10.18653/v1/2022.findings-acl.67.

Xiaojie Wang, Rui Zhang, Yu Sun, and Jianzhong Qi. KDGAN: knowledge distillation with generative adversarial networks. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages

783–794, 2018. URL https://proceedings.neurips.cc/paper/2018/hash/019d385eb67632a7e958e23f24bd07d7-Abstract.html.

Xinyu Wang, Yong Jiang, Nguyen Bach, Tao Wang, Zhongqiang Huang, Fei Huang, and Kewei Tu. Automated concatenation of embeddings for structured prediction. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli, editors, *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pages 2643–2660. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.acl-long.206. URL https://doi.org/10.18653/v1/2021.acl-long.206.

David Warde-Farley and Yoshua Bengio. Improving generative adversarial networks with denoising feature matching. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=S1X7nhsxl.

Thomas J. Watson. An empirical study of the naive bayes classifier. 2001.

Ralph Weischedel. Ontonotes release 5.0 ldc2013t19. Philadelphia: Linguistic Data Consortium, 2013. URL https://catalog.ldc.upenn.edu/LDC2013T19.

Ruobing Xie, Zhiyuan Liu, Jia Jia, Huanbo Luan, and Maosong Sun. Representation learning of knowledge graphs with entity descriptions. In Dale Schuurmans and Michael P. Wellman, editors, *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA*, pages 2659–2665. AAAI Press, 2016. URL http://www.aaai.org/ocs/index.php/AAAI/AAAI16/paper/view/12216.

Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. Representation learning on graphs with jumping knowledge networks. In Jennifer G. Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 5449–5458. PMLR, 2018. URL http://proceedings.mlr.press/v80/xu18c.html.

Xiangli Yang, Zixing Song, Irwin King, and Zenglin Xu. A survey on deep semi-supervised learning. *ArXiv*, abs/2103.00550, 2021.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. Xlnet: Generalized autoregressive pretraining for language understanding. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 5754–5764, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/dc6a7e655d7e5840e66733e9ee67cc69-Abstract.html.

Seung Hoon Yoo, Hui Geng, Tin Lok Chiu, Siu ki Yu, Dae Chul Cho, Jin Uk Heo, M S Choi, Il Hyun Choi, Cong Cung Van, N.V. Nhung, Byung Jun Min, and Ho Lee. Deep learning-based decision-tree classifier for covid-19 diagnosis from chest x-ray imaging. *Frontiers in Medicine*, 7, 2020.

Yue Yu, Simiao Zuo, Haoming Jiang, Wendi Ren, Tuo Zhao, and Chao Zhang. Fine-tuning pre-trained language model with weak supervision: A contrastive-regularized self-training approach. In Kristina Toutanova, Anna Rumshisky, Luke Zettlemoyer, Dilek Hakkani-Tür, Iz Beltagy, Steven Bethard, Ryan Cotterell, Tanmoy Chakraborty, and Yichao Zhou, editors, *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*, pages 1063–1077. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.naacl-main.84. URL https://doi.org/10.18653/v1/2021.naacl-main.84.

Man-Ching Yuen, Irwin King, and Kwong-Sak Leung. A survey of crowdsourcing systems. In *PASSAT/SocialCom 2011, Privacy, Security, Risk and Trust (PASSAT), 2011 IEEE Third International Conference on and 2011 IEEE Third International Conference on Social Computing (SocialCom), Boston, MA, USA, 9-11 Oct., 2011*, pages 766–773. IEEE Computer Society, 2011. doi: 10.1109/PASSAT/SocialCom.2011.203. URL https://doi.org/10.1109/PASSAT/SocialCom.2011.203.

Jieyu Zhang, Yue Yu, NameError, Yujing Wang, Yaming Yang, Mao Yang, and Alexander Ratner. WRENCH: A comprehensive benchmark for weak supervision. In Joaquin Vanschoren and Sai-Kit Yeung, editors, *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual*, 2021. URL https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/hash/1c9ac0159c94d8d0cbedc973445af2da-Abstract-round2.html.

Jieyu Zhang, Cheng-Yu Hsieh, Yue Yu, Chao Zhang, and Alexander Ratner. A survey on programmatic weak supervision. *CoRR*, abs/2202.05433, 2022. URL https://arxiv.org/abs/2202.05433.

Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In Corinna Cortes, Neil D. Lawrence, Daniel D. Lee, Masashi Sugiyama, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 649–657, 2015. URL https://proceedings.neurips.cc/paper/2015/hash/250cf8b51c773f3f8dc8b4be867a9a02-Abstract.html.

Yuhao Zhang, Victor Zhong, Danqi Chen, Gabor Angeli, and Christopher D. Manning. Position-aware attention and supervised data improve slot filling. In Martha Palmer, Rebecca Hwa, and Sebastian Riedel, editors, *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen,*

Denmark, September 9-11, 2017, pages 35–45. Association for Computational Linguistics, 2017. doi: 10.18653/v1/d17-1004. URL https://doi.org/10.18653/v1/d17-1004.

Zhengyan Zhang, Xu Han, Zhiyuan Liu, Xin Jiang, Maosong Sun, and Qun Liu. ERNIE: enhanced language representation with informative entities. In Anna Korhonen, David R. Traum, and Lluís Màrquez, editors, *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 1441–1451. Association for Computational Linguistics, 2019. doi: 10.18653/v1/p19-1139. URL https://doi.org/10.18653/v1/p19-1139.

Kang Zhao, Hua Xu, Yue Cheng, Xiaoteng Li, and Kai Gao. Representation iterative fusion based on heterogeneous graph neural network for joint entity and relation extraction. *Knowl. Based Syst.*, 219:106888, 2021. doi: 10.1016/j.knosys.2021.106888. URL https://doi.org/10.1016/j.knosys.2021.106888.

Peixiang Zhong, Di Wang, and Chunyan Miao. Knowledge-enriched transformer for emotion detection in textual conversations. In *EMNLP*, 2019.

Wenxuan Zhou and Muhao Chen. Learning from noisy labels for entity-centric information extraction. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih, editors, *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 5381–5392. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.emnlp-main.437. URL https://doi.org/10.18653/v1/2021.emnlp-main.437.