Chapter 5
Foundation Models for Information Extraction

Abstract In the chapter we consider Information Extraction approaches that automatically identify structured information in text documents and comprise a set of tasks. The Text Classification task assigns a document to one or more pre-defined content categories or classes. This includes many subtasks such as language identification, sentiment analysis, etc. The Word Sense Disambiguation task attaches a predefined meaning to each word in a document. The Named Entity Recognition task identifies named entities in a document. An entity is any object or concept mentioned in the text and a named entity is an entity that is referred to by a proper name. The Relation Extraction task aims to identify the relationship between entities extracted from a text. This covers many subtasks such as coreference resolution, entity linking, and event extraction. Most demanding is the joint extraction of entities and relations from a text. Traditionally, relatively small Pre-trained Language Models have been fine-tuned to these task and yield high performance, while larger Foundation Models achieve high scores with few-shot prompts, but usually have not been benchmarked.

Keywords Text classification · Named entity recognition · Relation extraction · Sentiment analysis · Language understanding

There are a large number of NLP applications of Pre-trained Language Models (PLMs), which can be roughly divided into three areas

- *Information Extraction* (*IE*) automatically identifies structured information in textual documents and analyzes language features (Chap. 5).
- *Natural Language Generation* (*NLG*) automatically generates new natural language text, often in response to some prompt (Chap. 6).
- *Multimodal Content Analysis* and generation integrates the understanding and production of content across two or more modalities like text, speech, image, video, etc (Chap. 7).

These applications are described in the three following chapters.

G. Paaß, S. Giesselbach, *Foundation Models for Natural Language Processing*,
Artificial Intelligence: Foundations, Theory, and Algorithms,
https://doi.org/10.1007/978-3-031-23190-2_5

Table 5.1 Language analysis tasks based on text classification illustrated by examples

Task	Description	Example
Language identification	Determine the language of a text, Sect. 1.2.	*Shakespeare lived 400 years ago* → *English*
Document classification	Assign a content category (class), e.g. economy, to a document or text, Sect. 5.1	*The Dow-Jones is up 50 points* → *economy*
Sentiment analysis	Classification of a text according to the sentiment expressed in it (e.g. positive, negative, neutral), Sect. 5.1	*Today I feel really lousy.* → *negative*
Hate speech detection	Recognize if a text contains hate speech, Sect. 5.1.1	*Immigrants infest our country* → *hate speech*
Fake news detection	Detect a text that contains fake news, Sect. 6.5.5	*Measles vaccination causes meningitis.* → *fake news*
Logical relation	Determine whether the second text contains a logical consequence, a contradiction, or a neutral statement relative to the first text, Sect. 2.1.5	*John has a flat.* $\leftrightarrow_{contradiction}$ *John is a homeless person.*
Text entailment	Does the first text imply the truth of the second text? Sect. 2.1.5	*Exercising improves health.* $\rightarrow_{entails}$ *Physical activity has good consequences.*
Paraphrase detection	Determine if two texts are semantically equivalent, Sect. 2.1.5	*Fred is tired. /Fred wants to sleep.* → *equivalent*
Dialog act classification	Determine the type of an utterance in a dialog (question, statement, request for action, etc.)	*Where is the dog?* → *question*

In the present chapter we focus on **information extraction** with PLMs. Information extraction includes the following tasks:

- *Text classification* assigns a document to one or more pre-defined content *categories* or classes (Sect. 5.1). Note that many subtasks can be formulated as classification problems, e.g. language identification, sentiment analysis, etc. (Table 5.1).
- *Word Sense Disambiguation* (*WSD*) connects a predefined meaning to each word in a document. This is especially important for the interpretation of *homonyms*, i.e. words that have several meanings depending on the context (Sect. 5.2).
- *Named Entity Recognition* (*NER*) identifies *named entities* in a document. An *entity* is an any object or concept mentioned in the text. A *named entity* is an entity that is referred to by a proper name. NER also associates a type with each entity, e.g. person, location, organization, etc. (Sect. 5.3).

- *Relation Extraction* aims to identify the relationship between *entities* extracted from a text (Sect. 5.4). This covers many subtasks such as coreference resolution, entity linking, and event extraction (Table 5.3).

Due to the large number of different approaches, we focus on representative models which exhibit a high performance at the time of writing. Traditionally relatively small PLMs have been fine-tuned to these task and yield high performance, while larger Foundation Models achieve high scores with few-shot prompts, but usually have not been benchmarked.

We outline the inner logic and main features of the methods, taking into account necessary resources, e.g. computational and memory requirements. For standard models a link to the description in earlier chapters is provided. Under the heading "Available Implementations" you will find links to available code and pre-trained models for a task. Good general sources for code are the websites [30, 35, 74, 79].

5.1 Text Classification

Automatic *text classification* is a common task in natural language processing where a *class*, (also called *category* or *label*) is assigned to a short text or a document. The set of classes is predefined and may contain just two classes (*binary classification*), or more classes (*multiclass classification*). Each text must be assigned a single class, which means that the classes are exclusive. Typical tasks include spam detection in emails, sentiment analysis , categorization of news articles , hate speech detection, dialog act classification, and many more. Some examples are listed in Table 5.1. Kowsari et al. [44], Li et al. [49] and Minaee et al. [64] provide surveys on text classification.

Often a document covers several topics simultaneously, e.g. a news article on the construction cost of a soccer stadium. In this case it is necessary to assign multiple classes to a document, in our example *"soccer"* and *"finance"*. This type of classification is called *multilabel classification*. *Extreme multilabel classification* is a variant containing a very large label set with thousands of labels.

There are a number of popular benchmarks to assess the performance of document classification approaches covering two or more classes. Typically, the benchmarks contain many thousand training and test examples. Table 5.2 describes the properties of some popular text classification benchmarks. Often documents are categorized according to the subjective opinions of users. An example are reviews of movies or restaurants, which can be classified as positive, negative, or neutral. Then the classification corresponds to a *sentiment analysis* task.

Early methods for document classification in the 1990s used classical machine learning approaches [44]. In the first preprocessing step, manually created features were extracted from the documents. In the second step, a classifier was trained with these features to reconstruct the manually assigned class labels (Sect. 1.3). Finally, this classifier was applied to new documents. Usually, *bag-of-words* representations were used to represent the input documents. Popular classification methods included

Table 5.2 Popular text classification benchmarks

Task	Description	Classes
IMDB [56]	Reviews from the movie rating page IMDB. 25k training, 25k test and 50k unlabeled reviews	Two classes: positive and negative
Yelp [131]	Yelp reviews of stores and restaurants. 560k training and 38k text reviews.	Binary: positive/negative multiclass: five star classes
DBpedia [7]	14 non-overlapping classes from the DBpedia ontology. Each class is represented by 40k training samples and 5k test samples,	14 different classes: company, artist, athlete, animal, album, film, etc.
ArXiv [32]	33k scientific articles from arXiv with documents of average length 6300 and length > 5000	11 classes: artificial intelligence, computer vision, group theory, etc.
SemEval-20 Task 12 [128]	14k Twitter tweets available for five languages: English, Arabic, Danish, Greek, Turkish	Two classes: offensive or not offensive.
EURLex-4K [53]	Benchmark of law documents containing 45,000 training examples with an average length of 727 words and an average of five correct classes per example	4271 non-exclusive classes
Amazon670k dataset [60]	Descriptions of amazon products. 490k training and 153k test samples. About 5.5 classes per document.	679k non-exclusive categories: products in the Amazon catalog, about 4 samples per category

naive Bayes, *logistic classifier*, the *support vector machine*, and tree-based methods like *random forests*. However, all these methods were hampered by the shortcomings of the bag-of-words representation (Sect. 1.3), which ignores the sequence of words in a document.

In the next sections, we consider current classification models for mutually exclusive as well as "overlapping" classes. It turns out that most of the current best approaches are based on PLMs.

5.1.1 Multiclass Classification with Exclusive Classes

A prominent application of **BERT** is fine-tuning for a classification task (Sect. 2.1.2). Here, a pre-trained BERT is adapted to this task by supervised fine-tuning, using the contextual embedding of the *"[CLS]"* token in the highest layer as input for a logistic classifier. This classifier is extremely successful for natural language understanding tasks (Sect. 2.1.5).

XLNet [120] is trained by reconstructing a permuted token sequence (Sect. 3.1.1), and is therefore able to capture a lot of knowledge about the language. It achieves 96.2% accuracy on the binary IMDB classification task. This performance is surpassed by **ERNIE-Doc** [26] with 97.1%. ERNIE-Doc is a transformer with an enhanced recurrence mechanism capable of considering many previous segments of a text in the same layer. The model aims to mitigate problems of other transformer-based models for long contexts such as the Longformer, which do not provide the contextual information of whole documents to each segment. The SOTA is currently held by a simpler model [101], which modifies the well known paragraph vector [47] and Naive Bayes weighted bag of n-grams. It achieves an accuracy of 97.4%.

The current best model on the IMDB dataset with 10 classes is **ALBERT-SEN** [23]. The authors propose an approach which evaluates the overall importance of sentences to the whole document, with the motivation that different sentences can contain different polarities but that the overall polarity depends on a few important sentences. Their model uses ALBERT (Sect. 3.1.1) to encode sentences via the *[SEP]* and *[CLS]* token representations. They concatenate these representations with class-weighted representations. Then they have a document encoder that calculates importance weights for every sentence and creates a weighted representation of the sentences as document representation. Finally, they calculate a sentiment score by utilizing the document representation and the class representations, which were also used in the sentence encoder. The model achieves an accuracy of 54.8%. It should be noted that subtle nuances in language expressions must be taken into account in this classification task with 10 classes.

For the Yelp benchmark, **XLNet** performs best for the binary version with an accuracy of 98.4% and achieves the second-best accuracy of 73.0% for the fine-granular version with 5 classes. The leading model for this task is **HAHNN** [1] with an accuracy of 73.3%. HAHNN combines convolutional layers, gated recurrent units and attention mechanisms. It builds on non-contextual FastText [16] embeddings as word representations and uses a stack of convolutional layers to obtain contextual information. This is followed by a word encoder which applies recurrent GRU cells to obtain word representations, and an attention mechanism to create weights for the input words. Sentence representations are then formed as an attention-weighted average of the words. Another GRU layer is employed to create sentence representations, which are then combined via attention to generate a document level representation. This establishes the input to a fully connected layer with softmax activation for classification.

BigBird [127] is especially valuable for classification tasks with long documents, as it can process input sequences of length 4096 (Sect. 3.2.1). Following BERT, the output embedding of the first *[CLS]* is input for the classifier. For the IMDB data with shorter documents there is no performance gain compared to simpler models. On the *ArXiv benchmark* [32] with documents of average length 6300 and 11 classes BigBird improves SOTA by about 5% points.

With the advent of Web 2.0 and the ability for users to create and share their own content with the world, the proliferation of harmful content such as hate

speech, has increased. This is now fueled by bots and machine learning models that automatically create such content at a scale that humans can barely manage. *Hate speech* is often defined as a hostile or disparaging communication by a person or group referring to characteristics such as race, color, national origin, gender, disability, religion, or sexual orientation [36]. According to European law, hate speech is a punishable criminal offense.

Hate speech detection can be solved as a text classification task. Recognizing such a text is difficult because the line between hate speech, irony, free speech, and art is blurred. Jahan et al. [36] and Yin et al. [123] give a systematic review on automatic hate speech detection. Because of the importance of the task, let's take a closer look at current approaches.

Roy et al. [88] follow a multilingual approach. They preprocess the text from Twitter by using a special tokenization of tweets. The cleaned text, emojis and segmented hashtags are encoded by different transformers and concatenated. A final multilayer perceptron generates the classification. The results for the *HASOC 2019 tweet dataset* [58] show that the additional signal from the emojis and the hashtags yield a performance boost for hate speech detection as well as for classifying the type of hate speech. They achieve F1-values of 90.3%, 81.9% and 75.5% on the English, German, and Hindi test sets.

Mathew et al. [59] argue that the decisions of hate speech classifiers should be explained. They present the *HateXplain* dataset with about 20k posts. The annotation contains class labels (hateful, offensive, or normal), the target group being vilified, and span annotations of words causing the classification. Overall a BERT model yields the best results in explaining the hate speech classification decisions.

A recent competition was the SemEval-20 Task 12 [128], where 14,100 Twitter tweets were manually labeled as either offensive or not offensive. Using a **RoBERTa** classifier (Sect. 3.1.1) Wiedemann et al. [110] achieved 92.0% F1-value and won the competition. In a later experiment an ensemble of *ALBERT* models (Sect. 3.1.1) increased this score to 92.6%. In summary, the automatic classification of hate speech can be solved by PLMs with high quality.

5.1.2 Multilabel Classification

Multilabel classification is required whenever a text can belong to multiple classes simultaneously. When a very large number of classes is available, this is sometimes called *extreme multilabel classification*. An example problem is the assignment of tags to Wikipedia articles, where Wikipedia has almost 500k tags. In multilabel classification usually a score or probability for each class is returned. This can be used to rank the classes. Traditional metrics such as accuracy, which assume that only one class is correct, cannot be applied. An alternative is to measure the quality of ranking induced by the score (c.f. Sect. 6.1.2). A popular measure for a predicted score vector $\hat{y}_i \in [0, 1]$ and a ground truth label vector $y_i \in \{0, 1\}$ is the *precision at*

k, which counts, how many correct classes are among the k classes with the highest score:

$$prec@k = \frac{1}{k} \sum_{l \in \text{rank}_k(\hat{y})} y_l \qquad DCG@k = \frac{1}{k} \sum_{l \in rank_k(\hat{y})} \frac{y_l}{\log(l+1)}, \tag{5.1}$$

where $\text{rank}(\hat{y}) = (i_1, \ldots, i_k)$ is the vector of the indices of the k largest values of $\hat{y}_i$ sorted in descending order $\hat{y}_{i_1} \geq \cdots \geq \hat{y}_{i_k}$. The second measure $DCG@k$ is the *discounted cumulative gain*, where the correct assignments y_l are weighted by their rank l transformed with $1/\log(l+1)$ [14]. This reflects that correct assignments with a lower rank should get a lower weight. In addition, there is a normalized version $nDCG@k$, where $DCG@k$ is divided by its maximal possible value.

Separate classifiers for each class often yield a very good accuracy, but suffer from very bad training and prediction time. In the worst case these classifiers have to be trained per label with all positive instances of a label and all instances of the other labels as negative samples. To mitigate this effect **Parabel** [83] is based on a tree ensemble. First Parabel creates label representations by averaging all the instances that belong to a label and normalizing this averaged vector to 1. Then balanced 2-means clustering is applied on the label space recursively until all leaf nodes in the clustered label tree contain fewer than M labels, e.g. $M = 100$. For each internal node of the tree and for the leaf nodes, classifiers are trained to decide which path of the tree an instance follows. Thus, a balanced label hierarchy is generated efficiently based on a label representation such that labels with similar inputs end up together at the leaves. Up to 3 such trees are used as an ensemble.

Finally, for each label, 1-vs-All classifiers are trained as a MAP estimate of the joint probability distribution over labels. The negative examples used for training these classifiers are drawn from the other labels in the same leaf, so the most similar or confusing counterexamples are employed. For prediction a beam search is performed in the tree and only for the k most probable labels a classification is actually performed. Parabel has been applied to problems with 7M labels and can make predictions in logarithmic time. Parabel is significantly faster at training and prediction than state-of-the-art extreme classifiers while having almost the same precision. On the EURLex-4K it achieves a prec@1 value of 81.5 and on the Amazon-670k a prec@1 value of 43.9, which is worse than the 45.4 of the best approach, but its time for prediction is only 1/1000.

AttentionXML [124] is a tree-based classifier, which uses contextual embeddings as input features. With an attention between the many labels and the tokens, AttentionXML represents a given text differently for each label. The architecture of AttentionXML consists of a word representation layer, a bidirectional LSTM layer, an attention layer with attention from all labels to the BiLSTM (Sect. 1.6) encoded input and lastly a fully connected layer and an output layer.

AttentionXML first builds a deep tree similar to Parabel. Then the tree is compressed to a shallow and wide tree, which allows to handle millions of categories, especially for "tail labels", i.e. classes with only a few examples in the

training set [37]. The model uses the binary cross-entropy loss function. For each level of the tree this model is trained, being initialized with the model of the prior tree level. AttentionXML trains label ranking with negative labels sampled by fine-tuned label recalling models. For prediction the tree is used for a beam search, so only tree branches where the parent nodes have highest scores are considered.

On the *EURLex-4K benchmark* AttentionXML achieves $prec@1 = 87.1\%$ and $prec@5 = 61.9\%$. This means that the highest scoring prediction of the model is correct for 87.1% of the test predictions and 61.9% of the five highest scoring predictions are correct. Note that the choice of k should be made according to the average number of labels per document in the training set. On the *Amazon670k dataset* [60] with 679k categories AttentionXML achieves $prec@1 = 47.6\%$ and $prec@5 = 38.9\%$. This means that about 40% of the alternative products are correctly identified.

LightXML [39] employs a transformer encoder to generate contextual word features and generates negative examples for each category in a dynamic way. First, a set of label clusters is created based on the input features so that each label belongs to one cluster. Then a pre-trained model like RoBERTa (Sect. 3.1.1) is employed to encode the input text of an instance into contextual embeddings. To represent the input text of a training example, the embeddings of the *[CLS]* token in the last five layers are concatenated.

A specific *label recalling* model aims to predict the label clusters using the *[CLS]* embeddings as input. In addition, the *label ranking model* receives the *[CLS]* embeddings of a training instance as well as the corresponding label . Negative examples with other labels are dynamically generated with the label recalling model. The loss terms of both the generator and the discriminator are combined in a joint loss function allowing end-to-end training. On the EURLex-4K benchmark LightXML achieves a $prec@1 = 87.6\%$ and $prec@5 = 63.4\%$. On the Amazon670k benchmark it reaches a $prec@1 = 49.1\%$ and $prec@5 = 39.6\%$. Both values are slightly better than those of AttentionXML. The approach also demonstrates SOTA performance compared to 7 alternative model on three other multilabel datasets.

Overlap [51] groups labels into overlapping clusters. In product categorization, for example, the tag *"belt"* can be related to a vehicle belt (in the *"vehicle accessories"* category), or a man's belt (under *"clothing"* category). Each label can now occur at most λ-times, where λ is a hyperparameter of the approach. The authors initialize their partitioning with a balanced k-means clustering and then proceed with an optimization method to reassign labels in a way that maximizes the precision rate. On the Amazon670k benchmark the model reaches SOTA values of $prec@1 = 50.7\%$ and $prec@5 = 41.6\%$. There are also alternative models with a tree-based search, which are able to increase recall rates and reduce effort [22].

There is a great similarity of extreme multilabel classification with text retrieval, which is covered in Sect. 6.1. This group of text applications has seen a large progress in recent years. For dense retrieval the query and the document representations are encoded by a BERT model, and the documents with largest cosine similarity are returned. Probably many approaches from this field may be used for text classification.

5.1.3 *Few- and Zero-Shot Classification*

Large autoregressive language models like GPT-2, GPT-3, Gopher and PaLM have acquired an enormous amount of information about facts and language by pre-training. They can be instructed to classify a text by a few examples [76], as described in Sect. 3.6.3. Figure 5.1 provides an example prompt for the classification of a text by sentiment [91]. This means that no additional fine-tuning dataset is required, but only a prompt with a few examples. In the same way the pre-trained Gopher model [85] was applied to a comprehensive set of about 150 benchmark tasks, which require the generation of answers using few-shot instructions. Similar to other autoregressive models it may predict class labels for documents (Sect. 2.2.5). As the results show [85, p. 56], Gopher is often able to outperform conventional PLMs fine-tuned on the domain. Therefore, classification by instruction seems to be a viable alternative, if a large autoregressive PLM such as GPT-3, Gopher or GPT-Neo is available.

Recently, the *RAFT* [3] benchmark was released. RAFT is specifically designed for evaluating few-shot performance in text classification tasks. It covers 11 real-world datasets, 8 of which are binary classification, two contain three classes, and one contains 77 classes. Each task comes with natural language instructions and 50 labeled training examples. An example benchmark is *"Label the sentence based on whether it is related to an adverse drug effect. Sentence: No regional side effects were noted. Label: not related. ..."*. A prompt contained less than 50 examples. The performance is measured by an average F1 over all 11 tasks. On these RAFT benchmarks BART yields an F1 average of 38.2%, GPT-Neo (2.7B) achieves 48.1%,

Prompt:
Tweet: "I hate it when my phone battery dies."
Sentiment: Negative
- - -
Tweet: "My day has been 👍"
Sentiment: Positive
- - -
Tweet: "This is the link to the article"
Sentiment: Neutral
- - -
Tweet: "This new music video was incredible"
Sentiment:

GPT-Neo:
Positive

Fig. 5.1 A query for few-shot learning for sentiment analysis with GPT-Neo, a free version of GPT with 2.7B parameters. The query can be evaluated on the API [91]

AdaBoost decision trees 51.4%, and GPT-3 (175B) scores 62.7%. Humans achieve an average F1 of 73.5%.

PET [90] asks users to specify one or more patterns that convert an input example x into a *cloze prompt* (Sect. 2.1.2) so that it can be processed by a masked language model like BERT. In addition, users must describe the meaning of all output classes. This is done with a "verbalizer" that assigns a natural language expression to each output y. Multiple verbalizers may be specified for the same data. An example is *"I really enjoyed this movie. It was [MASK]."* and *"I really enjoyed this movie. Question: Is this a positive movie review? Answer: [MASK]."* for the text *"I really enjoyed this movie"*. The PLM is then trained to maximize $p(y|x)$ for observed pairs. PET achieves a new state of the art on RAFT with an average F1 of 82.2% and performs close to nonexpert humans for 7 out of 11 tasks.

Foundation Models can also be used to generate new data for a text classification task. If, for example, input for a restaurant classification task is required, the model can be prompted to generate a new restaurant review for a specific label Sect. 3.6.6. In this way training data for fine-tuning a model can be created.

Available Implementations

- The code and trained parameters of many classical models like BigBird, XLNET, T5 are available at Hugging Face https://huggingface.co/transformers/.
- The LightXML model code is here https://github.com/kongds/LightXML.
- The code of PET can be found here https://github.com/timoschick/pet.

5.1.4 Summary

For document classification, a PLM that has been pre-trained with a large set of documents is usually fine-tuned to solve a specific classification task. Typically, the embedding of a particular token such as *[CLS]* is used as input to a logistic classifier. This setup has outperformed all previous bag-of-word classifiers such as the SVM. Specialized PLM variants like XLNET or ALBERT show a higher performance because of their more effective pre-training. For longer documents, suitable models like BigBird yield good results. Identifying hate speech can be considered as a classification task, where good results are achieved with standard models such as BERT and RoBERTa.

The situation is different for multi-label classification, where several categories can be correct for one document. Here, tree-like classifiers in combination with contextual embeddings show good results. By the tree a small number of candidate classes can be selected reducing the training and execution times. Extreme multi-label classifications, such as matching product descriptions to related product descriptions, are close to a document retrieval tasks and can benefit from techniques developed in this area, e.g. dense retrieval by DPR.

Large pre-trained autoregressive language models like GPT-3, Gopher and PaLM may be instructed by few-shot learning to solve classification tasks. Recent approaches achieve a performance close to humans. Not long ago an API has been released which allows to pre-train GPT-3 and adapt it to specific data and specific classification tasks (Sect. 3.6.2). A simpler alternative is InstructGPT, which can be easily directed to perform a classification, e.g. a sentiment analysis (Sect. 3.6.5). However, a formal evaluation of the performance of this approach is not yet available, as the model would have to process the training data.

While PLMs have achieved promising results on demanding benchmarks, most of these models are not interpretable. For example, why does a model arrive at a particular classification? Why does a model outperform another model on one dataset, but performs worse on other datasets? Although the mechanisms of attention and self-attention provide some insight into the associations that lead to a particular outcome, detailed investigation of the underlying behavior and dynamics of these models is still lacking (Sect. 2.4.5). A thorough understanding of the theoretical aspects of these models would lead to a better acceptance of the results.

5.2 Word Sense Disambiguation

In nearly all languages the same word may express different concepts. An example is the word *"set"*, which may be a verb, an adjective, or a noun and can be interpreted as 'a group of things', a 'scenery', a mathematical concept, a sports term, etc. The WordNet [62] lexical database lists 45 different senses for this word. *Word sense disambiguation* (*WSD*) aims to distinguish these different meanings and annotate each word with its sense. It can be treated as a classification task, where each word is assigned to a sense of a sense inventory such as WordNet. The contextual embeddings generated by PLMs offer a way to identify these meanings. Bevilacqua et al. [13] provide a recent survey of WSD approaches.

WSD can be used for a number of purposes. A traditional application is search, where the different senses of the same word are distinguished in the query. *Lexical substitution* [13] aims to replace a word or phrase in a text with another with nearly identical meaning.

5.2.1 Sense Inventories

WSD obviously depends on the definition of senses, which have to be assigned to the words. The main sense inventory for WSD in English is *WordNet* [62]. It consist of expert-made *synsets*, which are sets of synonymous words that represent a unique concept. A word can belong to multiple synsets denoting its different meanings. Version 3.0 of WordNet covers 147,306 words (or phrases) and 117,659 synsets. WordNet is also available for languages other than English through the *Open*

Multilingual WordNet project [17]. *Wikipedia* is another sense inventory often used for *Entity Linking* (Sect. 5.3.3), where a person, a concept or an entity represented by a Wikipedia page has to be linked to a given *mention* of the entity in a text. *BabelNet* [71] is a mixture of WordNet, Wikipedia and several other lexical resources, such as Wiktionary [111] and OmegaWiki [75]. It is highly multilingual covering more than 500 languages.

WordNet's sense inventory is often too fine-grained. For example, the noun *"star"* has eight meanings in WordNet. The two meanings referring to a *"celestial body"* distinguish only whether the star is visible from earth or not. Both meanings are translated in Spanish as *"estrella"*, so this sense distinction is useless for this translation. It has been shown that for many tasks more coarse-grained sense inventories are better [81].

The best WSD algorithms use PLMs pre-trained on large document corpora. Through fine-tuning, they are trained to assign senses from the available sense inventory. In some cases, nearest neighbor operations are employed to measure the distance between embeddings and determine the most appropriate sense.

5.2.2 Models

GlossBERT [33] employs a pre-trained BERT encoder. Its fine-tuning input is both the context sentence (where the word is used in the specific sense) and the *gloss* (a sentence defining the meaning of the word). GlossBERT is trained to predict whether the gloss correctly describes the use of the target word. The *SemCor3.0* [61] benchmark is annotated with WordNet senses. GlossBERT achieves a new SOTA of 77.0% F1 on this data.

EWISER [12] expresses WSD as a simple *Word annotation* task (Sect. 2.1.3), where a sense label is assigned to each word. It starts with an average of BERT embeddings for each word v_t from different contexts and transforms them with a linear layer and the *Swish* [86] activation function $f(x) = x \cdot \text{sigmoid}(\beta x)$. For each combination of a word and a part-of-speech a set $S(v_t)$ of possible word senses and hypernyms is determined similar to [78]. Then the approach computes probabilities that a word belongs to a synset in $S(v_t)$. By this approach the prediction takes into account which WordNet senses are possible for a word. It achieves a new SOTA of 80.1% on a combination of WSD benchmarks. This value is also an estimated upper bound on human inter-annotator agreement [69], showing that WSD is on par with humans. The paper lists the results for a number of alternative approaches. The **BEM** model [15] is a similar system yielding comparable accuracy. A detailed analysis of how PLMs (especially BERT) capture lexical ambiguity can be found in [52]. The authors show that the embedding space of BERT covers enough detail to distinguish word senses.

MuLaN [9] is based on a multilingual list $\mathcal{D}$ of *synsets* in different languages. For example, $\mathcal{D}$ may contain the synset corresponding to the *"fountain"* meaning of *"spring"*, which is expressed in different languages as *"Quelle*$_{DE}$*"*, *"spring*$_{EN}$*"*, *"fountain*$_{EN}$*"*, *"manantial*$_{ES}$*"*, *"brollador*$_{CAT}$*"*, *"source*$_{FR}$*"*, *"fonte*$_{IT}$*"*, and *"sorgente*$_{IT}$*"*. The semantic repositories *WordNet* [62] and *BabelNet* [71] are employed to create $\mathcal{D}$. MuLaN has the task to annotate an unlabeled corpus U in the target language with senses using a corpus L_{lab} in the source language (e.g. English) as input, which is annotated with senses from $\mathcal{D}$. This is done in the following steps:

- *Creating embeddings*: The multilingual mBERT (Sect. 3.3.1) trained on 104 languages is used to compute the embedding $emb(\sigma, w)$ of every word w in context σ in L_{lab}. If w is split into multiple tokens, their average is used. If w is a compound, first the tokens of each word within the compound are averaged and then the average over words is taken as representation for w.
- *Candidate production*: Then for each word w with embedding $emb(\sigma, w)$ in context σ from L_{lab} the nearest 1000 neighbors from the unlabeled corpus U are determined by *FAISS* [40]. As an example we select the text span $v =$*"correre"* from the context $\tau =$*"Mi hanno consigliato di andare a correre."* in L_{lab} as the closest candidate $emb(\tau, v)$ for the instance $w =$*"running"* from the sentence $\sigma =$*"I've seen her go running in the park."*.
- *Synset compatibility*: Subsequently, it is checked if the closest candidate word v is contained in a synset of w in $\mathcal{D}$. Otherwise it is discarded.
- *Backward compatibility*: Finally, the nearest neighbors of $emb(\tau, v)$ in context τ in L_{lab} are determined. (τ, v) is only retained, if its nearest neighbor list contains w.
- *Dataset generation*: After a number of additional filtering steps the final annotation of words in the target corpus U is performed.

As a labeled corpus L_{lab} a union of *SemCor* [63] and the *WordNet Glos Corpus* (*WNG*) [46] is used, which are annotated with senses. As unlabeled corpus U the Wikipedia is used for Italian, French, Spanish and German. When tested on *SemEval-13* [70] and *SemEval-15* [66], MuLaN is the best system to annotate words with senses in the four languages with F1-values above 80%. An important advantage of MuLaN is that it is able to transfer sense annotations from high-resource to low-resource languages.

Escher [8] reformulates WSD as a span prediction problem. The input to the model is a sentence with a target word and all its possible sense definitions. The output is a text span identifying the gloss expressing the target words most suitable meaning. As an example consider Fig. 5.2 with the input sentence *"<s> The bully had to <t> back down </t>. </s>"* where the target word is enclosed in *"<t>"* and *"</t>"*. Subsequently, two glosses are appended.

The span is predicted similar to Sect. 2.1.3 by separately computing the probability for the first and last token of the span covering the correct gloss. In the example the sentence *"Move backwards from a certain position."* is selected as span, which describes the correct sense. By lowering the prior probability of the most

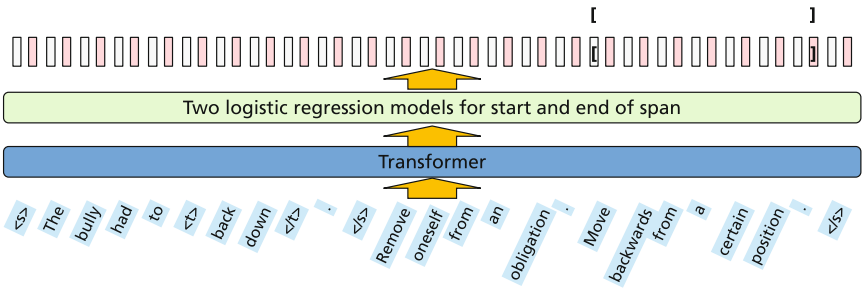

Fig. 5.2 Escher [8] takes as input a sentence, where the target word *"back down"* is enclosed by *"<t>"* and *"</t>"*. The most probable sense of the target word is indicated by the sentence selected by span prediction. A high probability of a span start is indicated by *"["* and a high probability of the span end is indicated by *"]"*

frequent sense for a word the approach is able to reduce the most frequent sense bias. Escher uses $\text{BART}_{\text{LARGE}}$ (Sect. 3.1.3) as PLM architecture, as it is effective for reading comprehension. The output of its last decoder layer is used to represent the input tokens and to compute the start and end token distributions. On a number of SemEval datasets [66] Escher has higher F1-scores compared to its competitors and this difference is statistically highly significant. Best results are achieved for nouns and adjectives with F1-values $> 83\%$, while for verbs the F1-value is only 69.3%.

ConSec [10] determines the sense of a token by considering not only the context words, but also the senses assigned to the neighboring words. It is based on an extension of DeBERTa, a BERT variant with superior performance (Sect. 3.1.1). ConSec uses WordNet example sentences with annotated meanings (glosses) as additional training data. The approach yields a SOTA of 83.2% F1 when applied to the *SemCor3.0* benchmark [61].

Available Implementations

- The codes of GlossBERT and EWISER and trained models are available for a number of different languages https://github.com/HSLCY/GlossBERT https://github.com/SapienzaNLP/ewiser.
- Escher along with the necessary training data is available at https://github.com/SapienzaNLP/esc.

5.2.3 Summary

WSD can be handled as a classification task, where each word is assigned to a number of possible meaning classes. Often WordNet is used as the sense inventory.

GlossBERT compares the contextual embedding of a word with the embedding of a word in an example sentence (gloss) of WordNet. EWISER and MULAN directly work on the synsets of WordNet and capture the sets of possible senses and hypernyms. They are able to annotate senses in four languages with an F1-value above 80%. Escher reformulates WSD as a span prediction problem increasing F1 to 83%. ConSec takes into account the senses of nearby tokens and achieves a similar performance.

As WSD models get better, there is a need for more demanding benchmark datasets, which possibly may be generated by adversarial techniques. Moreover, there is a trend to WSD models which are more robust to domain shift and can cope with text from social media documents. To advance WSD it is necessary to extend sense-annotated data, especially for rare senses. In addition, multilingual WSD systems may be constructed which require large-scale multilingual WSD benchmarks. There are tendencies in WSD to do away with the fixed sense inventory and to distinguish the senses in other ways, e.g., in a lexical substitution task or by generating the definition of a word in a particular context.

An opportunity is the integration of WSD with entity linking (Sect. 5.3.3), where the model is required to associate mentions with entries in a knowledge base such as Wikipedia. As WSD systems work fairly well now, it would be possible to combine them with other applications like question answering or dialog systems. It has to be tested, whether an explicit inclusion of WSD is able to generate better results. For retrieval tasks, WSD has been superseded by embedding-based methods (Sect. 6.1), which provide a better hit rate.

5.3 Named Entity Recognition

Named entity recognition (*NER*) refers to the task of tagging *mentions* of *named entities*, such as persons, organizations and locations in texts. Labeled datasets for NER exist across many domains, e.g. news, science and medicine [72]. Typically these datasets are annotated in the *IOB2 format*, which, for instance annotates the first token of a person with B-per and all other tokens of that entity with I-per. The O-tag is used for all tokens outside of entity mentions. An example is *"U.N.*$_{\text{B-org}}$ *official*$_{O}$ *Peter*$_{\text{B-per}}$ *Ekeus*$_{\text{I-per}}$ *heads*$_{O}$ *for*$_{O}$ *Bagdad*$_{\text{B-loc}}$*."* NER involves the prediction of these tags for each token, i.e. the suffixes in the prior example. Therefore, it can be considered as a classification task, where a tag is assigned to each token. A standard dataset for NER is the CoNLL-2003 dataset [89], which contains English resp. German news texts with annotations for persons, organizations, locations, and miscellaneous names. Surveys on NER are provided by Li et al. [48], Nasar et al. [68] and Bose et al. [18].

NER is particularly useful in areas with a highly specialized vocabulary. Examples include the fields of healthcare or electromobility, where many thousands of publications are released each year. Since few experts understand the terminology,

NER systems are particularly valuable for identifying publications on specialized topics. Of course, the NER types must be adapted to each area.

In the following section, we present approaches to ordinary NER where each word can have a single entity type. Named entities can also be nested, e.g. *"[[UK]$_{gpe}$ Embassy in [France]$_{gpe}$]$_{facility}$"*. This case is discussed in the second section. Even more challenging is the mapping of a named-entity phrase to the underlying unique entity in a knowledge base or ontology, e.g., a person. This is called entity linking and is discussed in the third section.

5.3.1 Flat Named Entity Recognition

In *flat named entity recognition* each token corresponds to at most one named entity. **BERT** can be fine-tuned to NER by predicting tags for each token using a logistic classifier (Fig. 2.5) as a final layer. For this setup $\text{BERT}_{\text{LARGE}}$ yielded 92.8% F1-value on the CoNLL-2003 test data. While the F1-values for persons and locations were higher ($\approx$ 95%), the F1-value for miscellaneous names (78%) was much lower, as these entities form a vaguely defined class.

LUKE [117] treats words and entities in a given text as independent objects, and outputs contextual embeddings of tokens and entities. The model is based on RoBERTa and trained to predict randomly masked words and entities in a large entity-annotated corpus derived from Wikipedia. In this way, it obtains a lot of information on the relation between entities in the text. It contains an entity-aware self-attention mechanism that is an extension of BERT's self-attention mechanism and takes into account embeddings, which indicate if a token represents text or an entity. It yields an F1-value of 94.3-F1 for CoNLL-2003, which is near-SOTA.

ACE [106] builds on the assumption that weighted sums $\sum_{i \in I} A_i * emb(v_i)$ of different embeddings $emb(v_i)$ of tokens v_i yield better results than single embeddings. A controller samples a subset I from a set of eight embeddings (e.g. $\text{BERT}_{\text{BASE}}$, GloVe, fastText, etc.) and a NER model is trained and returns an accuracy score. The accuracy is treated as a reward signal in a reinforcement setting using the policy gradient algorithm ([112]) to select an optimal subset I. As NER model a BiLSTM model (Sect. 1.6) with a final CRF-layer was chosen. A CRF (Conditional Random Field) [100] is able to model the probabilistic relation between the tags in detail. The fine-tuned model reaches a SOTA F1-score of 94.6% for CoNLL-2003.

KeBioLM [126] is a biomedical pre-trained language model aiming to improve NER by including additional knowledge. The authors extract 660M entities from the *PubMed corpus* [73] with abstracts of biomedical literature and link them to the UMLS knowledge base that contains more than 4M entities and their synonyms as well as relations. They train a variant of BERT on the PubMed data and explicitly generate embeddings for entities. Relation information is included by the TransE-mechanism (Sect. 3.4.1). The joint loss function is a mixture of loss functions for masked language modeling, entity detection, and entity linking. The *JNLPBA*

benchmark contains 2000 PubMed abstracts with molecular biology-related entities. KeBioLM reaches a SOTA of 82.0% F1 on JNLPBA. This shows that pre-training on domain texts and the inclusion of additional knowledge can improve NER results.

Retrieval is a way to enhance the context a PLM may use for NER. Wang et al. [107] query a search engine with the input text that should be tagged. They rank (Sect. 3.4.5) the returned results by the similarity of RoBERTa embeddings and concatenate the top ranked results and the input text. This is fed into a variant of RoBERTa to generate token embeddings. As the model can exploit the attention to the retrieved texts, the generated embeddings are potentially more expressive. The results on CoNLL 2003 indicate that retrieval can increase the F1-value about 0.5% and could be combined with current SOTA-models.

5.3.2 Nested Named Entity Recognition

Often named entities have an internal structure. An example for such *nested entities* is the sentence *"Last night, the [[Chinese]$_{gpe}$ embassy in [France]$_{gpe}$]$_{facility}$ was closed."* In this case a single token may have several entity tags and the NER task has to be formulated differently.

MRC [50] treats nested NER as a question-answering task. For example, the extraction of entities with a "location" label is formalized as the question: *"Which locations are mentioned in the text?"* The questions are formulated using templates that reflect the annotation guidelines. When these questions are answered for each entity type, overlapping named entities can be detected. MRC uses BERT's span prediction approach (Sect. 2.1.3) to mark the beginning and end of spans in the token sequence for an entity type. In addition, MRC predicts the start and the end of each entity to allow that there are overlapping entities of the same type.

Nested entities are common in the medical domain. The *Genia Corpus* [43] contains entity annotations for proteins, viruses, DNA, RNA and many more, with 17% of the entities being nested. MRC achieves a SOTA of 83.8% F1 on the Genia benchmark. The ACE-2005 benchmark [104] contain diverse nested entities like persons, facilities, or vehicles with an overlap of 22%. MRC reached an F1-value of 86.9% for ACE-2005. A similar approach [125] also predicts spans of different entities and yields 85.4% for ACE-2005. A two-stage algorithm called Locate and Label is proposed by Shen et al. [93], who first extract candidate entities and then categorize them in a second step. They yield 86.7% for the nested NER on ACE-2005 using BERT or one of its variants.

Instead of using a BERT model pre-trained on general documents, **PubMedBERT** [102] pre-trains its BERT model with 100M parameters exclusively on 21 GB medical texts from PubMed. PubMedBERT achieves 86.3% F1 for NER on the *BLURB benchmark* [31]. The model also yields SOTA scores for other task like classification and relation extraction summarized in an average score of 82.9%. This result strongly supports pre-training on domain-specific data. **BioELECTRA** [42] is a biomedical domain-specific language encoder model that adapts ELECTRA

(Sect. 3.1.1) for the Biomedical domain. ELECTRA employs a sample-efficient 'replaced token detection' technique for pre-training, which causes the model to include an enormous amount of information from the training data. BioELECTRA is pre-trained on PubMed and PubMed Central full-text medical articles. For NER, it arrives at the best score with 86.7% F1-value on the BLURB benchmark [31]. The model also yields a similar score of 82.6% as PubMedBERT for the other BLURB tasks.

Available Implementations

- $\text{BERT}_{\text{LARGE}}$ for token classification https://huggingface.co/transformers/model_doc/model_doc/bert.html,
- Luke https://huggingface.co/transformers/model_doc/model_doc/luke.html
- ACE https://github.com/Alibaba-NLP/ACE,
- MRC https://github.com/ShannonAI/mrc-for-flat-nested-ner
- Locate and Label [93] https://github.com/tricktreat/locate-and-label
- Bioelectra for nested NER https://github.com/kamalkraj/BioELECTRA

5.3.3 Entity Linking

After identifying a named entity in a text (*entity mention*), one often wants to disambiguate it, i.e. assign the mention to a unique entity in a KB or ontology. This involves unifying different writings of an entity name. To attach the corresponding facts and relation to the same entity, it is important to link the different writings of a name, e.g. *"Joe Biden was elected as 46th president of the United States of America"* and *"President Biden was born in Scranton Pennsylvania"*. Note that there exist about 35 writings for the name *"Muammar Muhammad Abu Minyar al-Gaddafi"*, e.g. *"Qadhafi"*, *"Gaddafi"* and *"Gadhafi"* in addition to versions with the different first names. *Entity Linking* approaches aim to solve this problem.

Entity linking is useful for tasks such as knowledge base population, chatbots, recommender systems, and question answering to identify the correct object or entity referred to. It is also required as a preprocessing step for models that need the entity identity, such as KnowBERT [80] or ERNIE [99] (Sect. 3.4.1). Early approaches rely on semantic embeddings to match entity mentions belonging together [82]. Modern procedures use contextual embeddings to characterize the entity mentions. Sevgili et al. [92] provide a comprehensive survey of Deep Learning based entity linking approaches. They sketch the general solution architecture of entity linking approaches as shown in Fig. 5.3 and compare different methods.

BLINK [113] follows the scheme of Fig. 5.3. First entity mentions together with their types are extracted from a text by NER. Then it uses a BERT model to compute embeddings for mention contexts and the entity descriptions in the KB. This also involves the normalization of entity names. Using an efficient approximate

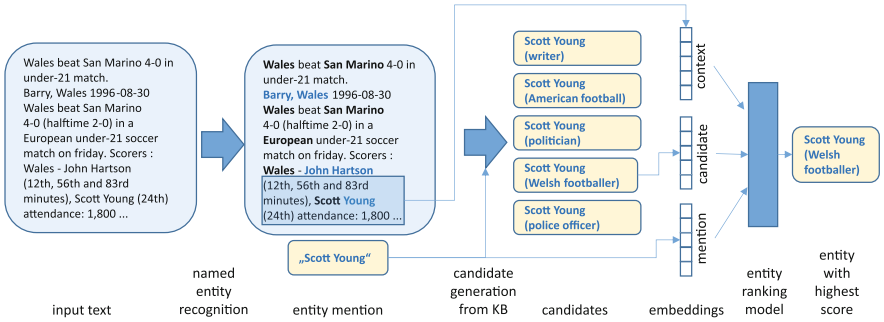

Fig. 5.3 Entity Linking includes the three steps entity recognition, which identifies entity mentions in a text, candidate generation generating possible entities for the mention using the KB, and entity ranking, computing a similarity score between the candidates and the mention. Image adapted from [92], reprinted with kind permission of authors

k-nearest-neighbor indexing scheme FAISS [40] for embeddings (Sect. 6.1.4). FAISS is able to retrieve the best matching entity candidates from the KB with little computational effort. This approach is identical to dense retrieval by DPR (Sect. 3.4.5). Each retrieved candidate is then examined more carefully with a cross-encoder that concatenates the input context, the mention and entity text and assigns a score to each candidate entity. Finally, the candidate with the highest score is selected. Although no explicit entity embeddings are computed, the approach achieves SOTA on the *TACKBP-2010 benchmark* [29] with an accuracy of 94.5%. A very similar approach is chosen by **EntQA** [130], which also exploits a retriever-reader architecture and yields competitive results on several benchmarks.

GENRE [25] departs from the common solution architecture to most entity linking approaches and uses the encoder-decoder model BART (Sect. 3.1.3) to disambiguate entities. This model has to recover text corrupted by a number of different approaches during pre-training and therefore gathers a lot of knowledge about language. The model is fine-tuned to generate disambiguated named entities. For example, the sentence *"In 1503, Leonardo began painting the Mona Lisa."* is translated to *"In 1503, [Leonardo](Leonardo da Vinci) began painting the Mona Lisa."*, where *"[Leonardo](Leonardo da Vinci)"* and *"Mona Lisa"* are the unique headings of the corresponding articles in Wikipedia. GENRE uses a constrained BEAM search for decoding, which either copies the input text or generates a unique Wikipedia entity name. In addition, GENRE can perform mention detection and end-to-end entity linking by associating a mention with the corresponding KB entity (e.g. the Wikipedia article). On six different benchmarks, GENRE achieves an average F1-value of 88.8% and outperforming BLINK, which scores 77.0%. In addition, GENRE has a smaller memory footprint (2.1 GB) than BLINK (30.1 GB). Finally, the model has a tendency to copy the mention exactly, which is helpful for new, unseen named entities.

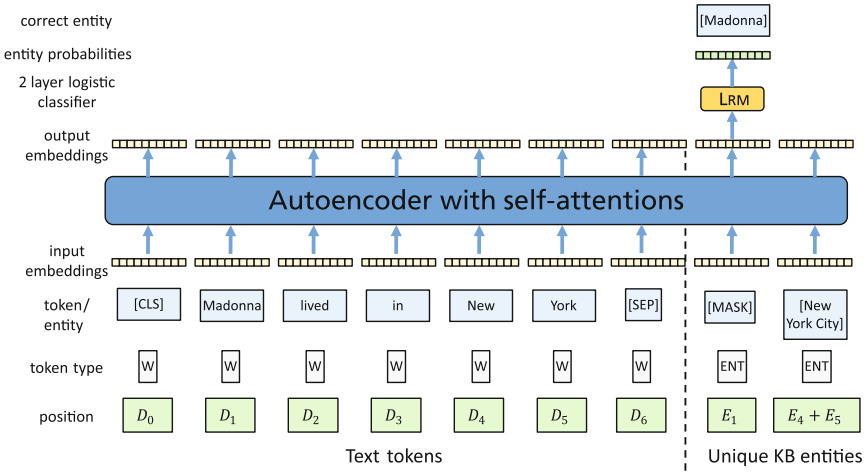

Fig. 5.4 $\text{BERT}_{\text{LARGE}}$ can be fine-tuned to predict masked 'entity tokens' taking into account the corresponding text. During application successively the entities with highest probability are assigned. In this way, the joint probability of entities can be exploited [118]

EntMask [118] is similar to LUKE (Sect. 3.4.4) and learns to predict masked entities. To disambiguate new mentions, the authors use local contextual information based on words, and global contextual information based on already disambiguated entities. Their model is trained to jointly produce embeddings of words and entities and is also based on $\text{BERT}_{\text{LARGE}}$. For fine-tuning 30% entities corresponding to Wikipedia hyperlinks are masked randomly and have to be predicted as shown in Fig. 5.4. During application the model predicts an entity for each mention, and from the unresolved mentions actually assigns the mention with the highest probability as 'observed'. In this way, this assignment can influence the prediction for the remaining mentions, introducing a global perspective. On a number of benchmarks the approach yields roughly similar results to GENRE, with a small advantage on a few benchmarks.

Available Implementations

- GENRE: model source code and datasets from Facebook https://github.com/facebookresearch/GENRE
- BLINK available at https://github.com/facebookresearch/BLINK
- EntMask code: https://github.com/studio-ousia/luke.

5.3.4 Summary

It is well known that named entities play a crucial role in understanding the meaning of a text. Thousands of new named entities appear every day, requiring special effort to interpret their sense. Due to the availability of contextual embeddings in PLMs Named Entity Recognition (NER) could increase F1-value on the CoNLL 2003 benchmark from 85% to 94.6%, dramatically reducing errors. The standard approach is token annotation by BERT, which marks each token with its corresponding entity type. Higher performance can be achieved by treating named entities as special tokens (LUKE), combining different kinds of embeddings (ACE), or using retrieval approaches based on embeddings. Empirical evaluations demonstrate that it is extremely important to train the underlying PLM on domain texts, e.g. from the medical domain. Single tokens or compounds can belong to multiple entity types at the same time. For this, nested NER question-answering approaches can be used to mark token spans as belonging to an entity type. Again training on domain texts is essential.

In Sect. 5.4.4 approaches for joint entity and relation extraction are presented. The approaches described there can also be used for NER alone and promise high performance. An example is REBEL, which uses the BART encoder-decoder to translate the input sentence to a unique representation of the covered entities and relations.

Entity linking aims to map an entity mention to the underlying unique entity in a KB. One approach exploits the retriever-reader architecture to find entity candidates from a knowledge base (BLINK, EntQA). Subsequently, a reader module scrutinizes candidates and the mention to arrive at a final assignment. An alternative is GENRE's encoder-decoder architecture, which translates entity mentions to unique entity names. Finally, a BERT model can determine self-attentions between token embeddings and entity embeddings and exploit this to predict unique entities contained in a text.

The majority of entity linking models still rely on external knowledge like Wikipedia for the candidate generation step. However, this is not sufficient when identifying a person who is not a celebrity. In this case we have to perform a search in the web or social media to find information. As retrieval-reader approaches gain popularity, this may be possible in the future. It turns out that NER and entity linking should be performed jointly, i.e. assignments should take into account each other to increase accuracy.

5.4 Relation Extraction

After identifying relevant entities in a sentence, a crucial part of information extraction is often the extraction and classification of relations between these entities. This is useful, for example, when we automatically want to populate databases

Table 5.3 Language analysis tasks based on relation extraction [4, p. 10]. Underlining indicates phrases annotated by the model

Task	Description	Example
Coreference resolution	Group phrases which refer to the same object.	*Betty*$_{(1)}$ *loves her*$_{(1)}$ *cute dog*$_{(2)}$.
Aspect-based sentiment analysis	Extract phrases (aspects) from a text and determine sentiments for them (positive, negative, neutral).	*The steak*$_{aspect}$ *was horrible*$_{negative}$.
Entity relation extraction	Extract relations among entities or concepts in a text.	*Peter works as a lawyer.* $\rightarrow$ *profession(Peter, lawyer)*
Event extraction	Extract events, i.e. n-ary relations among entities or nouns in a text.	*At noon*$_{time}$ *terrorists*$_{attacker}$ *detonated a bomb*$_{instrument}$ *in Paris*$_{place}$. $\rightarrow$ *conflict-attack*
Semantic role labeling	For each verb determine the role of phrases w.r. to the verb.	*Mary*$_{agent}$ *sold*$_{verb}$ *the book*$_{theme}$ *to John*$_{recipient}$.

or knowledge graphs with linked information. Table 5.3 contains examples of language analysis tasks based on relation extraction that are discussed in this section. Instances include coreference resolution, i.e. finding different mentions of an entity in the same text, aspect-based sentiment analysis, which links phrases in a text to opinions about them, or semantic role labeling, which identifies the function of a phrase for a predicate in a sentence. Because entity linking associates mentions of entities with the underlying unique object or person in an ontology, it differs from relation extraction. A survey on prior work in relation extraction is given by Nasar et al. [68].

5.4.1 Coreference Resolution

A first type of relation extraction is *coreference resolution*, whose goal is to establish a relation between all entity mentions in a text that refer to the same real-world entities. As an example, consider the sentence *"I voted for Biden because he was most aligned with my values", she said.* where *"I"*, *"my"*, and *"she"* refer to the speaker, and *"Biden"* and *"he"* pertain to Joe Biden. Due to the combinatorial number of subsets of related phrases, coreference analysis is one of the most challenging tasks of NLP. A survey of coreference resolution is provided by Stylianou et al. [98].

SpanBERT [41] is a version of BERT, which predicts contiguous subsequences of masked tokens during pre-training, and therefore accumulates knowledge about spans of words (Sect. 3.1.1). The authors consider all possible spans of text and identify relevant mentions spans. In parallel, for each span x, the preceding spans y

are examined, and a scoring function estimates whether the spans refer to the same entity.

This scoring function is defined as $s(x, y) = s_m(x) + s_m(y) + s_c(x, y)$. Here $s_m(x)$ and $s_m(y)$ measure how likely x and y are entity mentions. $s_c(x, y)$ determines how likely x and y refer to the same entity. As input from a span, the scoring function gets the output embeddings of the two span endpoints and a summary of the tokens embeddings of the span. The probability that y is coreferent to x is computed as $p(y) = \exp(s(x, y))/\sum_{y' \in Y} \exp(s(x, y'))$. In this way, subsets of spans mentioning the same entity are formed. During the iterations of the approach, the span definitions may be refined, and an antecedent pruning mechanism is applied to reduce the number of spans to be considered. *OntoNotes* [109] is a corpus of 1.5M words comprising various genres of text with structural information, e.g. coreference. After fine-tuning on OntoNotes, Span-BERT achieves a SOTA result of 79.6% F1-value on the test set. Dobrovolskii [27] propose a variant which performs its analysis on the word level thus reducing the complexity of the task. It raises the SOTA on OntoNotes to 81.0%.

CorefQA [114] solves coreference resolution as a question-answering problem. A first stage considers all spans up to a maximum length as potential mentions. The authors use a SpanBERT model to compute embeddings for all tokens. To reduce the number of mentions, a proposal module combining the start and end embeddings of spans is pre-trained to predict relevant mentions. Subsequently, each mention is in turn surrounded by special tokens and the network is trained to mark all coreferent spans similar to the question-answering fine-tuning of BERT (Sect. 2.1.3). To reduce the number of computations only a limited number of candidates in one direction is considered. The mention proposal and mention clustering can be trained end-to-end. On the coreference benchmark CoNLL 2012 [84] the approach improves SOTA significantly to 83.1% F1-value. Toshniwal et al. [103] extend this approach by tracking only a small bounded number of entities at a time. This approach can reach a high accuracy in coreference resolution even for long documents.

Available Implementations

- SpanBERT for relation extraction and coreference resolution at GitHub https://github.com/facebookresearch/SpanBERT
- CorefQA at GitHub https://github.com/ShannonAI/CorefQA

5.4.2 *Sentence-Level Relation Extraction*

There are various types of relations which can be extracted, e.g. in the sentence *"Goethe succumbed to his suffering in Weimar"* the *"died-in"* relation relates a person (*"Goethe"*) to a location (*"Weimar"*). In this section we assume that entities

have already been extracted from a sentence by NER (Sect. 5.3). Therefore, NER errors will increase the errors for relation extraction.

SpanBERT [41] is particularly suitable for relation extraction, since entity mentions often span over multiple tokens, and are masked by SpanBERT during pre-training (Sect. 3.1.1). For fine-tuning the model gets one sentence and two spans with possible relation arguments as input, which are replaced by their NER tags. An example is *"[CLS] [SUBJ-PER] was born in [OBJ-LOC] , Michigan, . . ."*. The final [CLS] embedding is input to a logistic classifier, which predicts one of the 42 predefined relation types, including "no relation". *Re-TACRED* [97] is a large-scale relation extraction dataset with 120k examples covering 41 relation types (e.g., per:schools-attended and org:members) and carefully checked relation annotations. SpanBERT showed good performance on Re-TACRED with 85.3% F1-value [95].

RoBERTa (Sect. 3.1.1) can be used to generate token embeddings for relation extraction. Zhou et al. [135] evaluate various entity representation techniques. They use $\text{RoBERTa}_{\text{LARGE}}$ to encode the input text by embeddings of the last layer. The embeddings of the first token in each span of relation argument mentions are used to represent these arguments. These are concatenated and adopted as input for a softmax classifier. It turns out that enclosing an entity and adding its type with special tokens yields the best results on the Re-TACRED dataset with 91.1% F1-value.

Relation-QA [24] rephrase the relation classification problem into a question answering problem. Consider the sentence s = *"Sam Brown was born in 1991."* with the extracted entities *"Sam Brown"* and *"1991"*. Then the authors create two queries, such as *"When was Sam Brown born?"* and *"Who was born in 1991?"*. They fine-tune ALBERT (Sect. 3.1.1) to answer these queries by marking the spans containing the desired entity. If no span is returned the relation does not hold. The approach achieves an F1-value of 74.8% for TACRED, an older version of ReTACRED with many annotation problems. **RECENT** [55] extends SpanBERT and trains more than one relation classification model, i.e. one classifier for each different pair of entity types. This restricts the possible output relation types and helps to increase performance. On TACRED the approach yields a SOTA F1-value of 75.2%.

5.4.3 Document-Level Relation Extraction

Especially for larger documents, the assumption that relations occur only inside a sentence is too restrictive. Therefore, some models check for relations on the document level. When relation arguments are in different sentences the corresponding entities are often only referred to via coreferent mentions. Therefore, we assume in this section that entities have been extracted and grouped into clusters denoting the same entity by coreference resolution (Sect. 5.4.1). Obviously the errors of coreference resolution will increase the final relation extraction errors.

SSAN [115] (Structured Self-Attention Network) directly takes into account structural information such as coreference and cooccurrence of entity mentions for PLMs such as RoBERTa. The authors modify the self-attention computations in encoder blocks by adding specific biases, if two mentions refer to the same entity and/or are located in the same sentence. These biases are computed from the query and key vectors by a "transformation model" trained during fine-tuning. Therefore, the scalar products between keys and queries are modified depending on whether the corresponding tokens are coreferent, in the same sentence, or not. Entity embeddings are obtained via average pooling of token embeddings of the entity mention. For each pair emb_i, emb_j of entity embeddings the probability of a relation r is computed by a bilinear transformation $\text{sigmoid}(emb_i^\intercal W_r emb_j)$ with a trainable parameter matrix W_r.

DocRED [121] is a large benchmark of documents annotated with named entities, coreferences, and relations whose arguments may be located in different sentences. Using $\text{RoBERTa}_{\text{LARGE}}$ as base network, the authors achieve a SOTA of 65.9% F1 on DocRED. Using a special BERT version SciBERT [11] trained on scientific papers from Semantic Scholar, the algorithm also yields SOTA results for benchmarks with chemical as well as biological texts.

ATLOP [136] marks the start and end of a mentions by a special token and encodes a document by BERT resulting in embeddings for each token. The embedding of token at the mention start is used as the mention embeddings. An entity embedding is computed by pooling coreferent mentions. The first and the second argument entity embedding of a relation are transformed by different fully connected layers to $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$. Subsequently, the probability of a relation r for an entity pair is estimated by a sparse bilinear transformation $\text{sigmoid}(\boldsymbol{x}_1^\intercal W \boldsymbol{x}_2)$. Trainable probability thresholds are used to decide if a relation holds. On the DocRED benchmark the model achieves an F1-value of 63.4%.

5.4.4 Joint Entity and Relation Extraction

Since NER and relation extraction are closely related tasks and relation extraction depends on the results of NER, it is a natural choice to model these tasks jointly.

UniRE [108] encodes entity and relation properties in a joint matrix, which has a row and a column for each text token. While named entities, e.g. PER, are marked on the diagonal, relations are matrix entries off-diagonal. If, for example, *"David Perkins"* lives in *"California"* the matrix entries in the rows of the *"David Perkins"* tokens and the columns of the *"California"* tokens are marked with the $PHYS$ relation. Note that in this way asymmetric relations may be specified.

All words in the input are encoded using a BERT encoder and then a biaffine model is used to create a scoring vector for a pair h_i and h_j of embeddings

$$p(y_{i,j}|s) = \text{softmax}\left((\boldsymbol{h}_i^{first})^\intercal U_1 \boldsymbol{h}_j^{sec} + U_2[\boldsymbol{h}_i^{first}, \boldsymbol{h}_j^{sec}] + b\right), \tag{5.2}$$

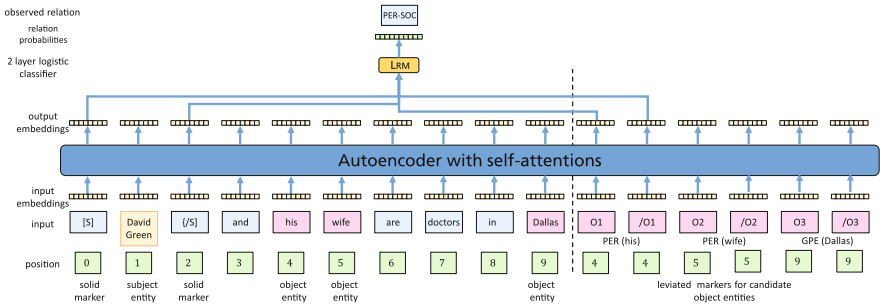

Fig. 5.5 For a possible relation the PL-marker model marks the first relation argument by special 'solid' markers and the possible second arguments by 'leviated' markers outside the text. The latter get the same positions as the corresponding tokens, and do not influence the embeddings of normal tokens during attention computation. The marker embeddings are concatenated to compute the probability of the corresponding relation [122]

where $\boldsymbol{h}_i^{first} = \text{FCL}_{first}(\boldsymbol{h}_i)$ and $\boldsymbol{h}_i^{sec} = \text{FCL}_{sec}(\boldsymbol{h}_i)$ are fully connected layer transformations of the first and second relation argument respectively. The softmax function obtains a probability distribution over the entity and relation labels for all matrix cells. The model minimizes three losses, one based on the actual labels of each cell, one based on the knowledge that diagonal of entity labels should be symmetrical and one based on the fact that a relation label implies that respective entity labels must be present. *ACE 2005* [104] consists of text of various types annotated for entities, relations and events. On ACE 2005 UniRE yields an F1-value of 66.0% for joint entity and relation extraction, which is less than the current SOTA of 70.5%.

PL-Marker [122] investigate different types of mention encodings. For a possible relation it surrounds the first argument span (subject) by solid marker tokens. The possible second argument spans (objects) are marked by *leviated tokens* Oi and $/Oi$ outside the text (Fig. 5.5). These get the same position embeddings as the corresponding object spans in the text. Their attention connections are restricted, i.e they are visible to each other, but not to the text token and other pairs of markers. Therefore, depending on the subject span the object token embeddings can capture different aspects. For each pair of subject-object arguments, the corresponding embeddings are concatenated and used as input to a logistic classifier to estimate the probability of the possible relations (or 'no relation'). Pre-trained variants of BERT are fine-tuned with ACE 2005 to predict the relations. With a $\text{BERT}_{\text{BASE}}$ model of 105M parameters the approach yields an F1-value of 68.8% on the ACE05 benchmark. If $\text{ALBERT}_{\text{XXLARGE}}$ [45] with 235M parameters is used to compute the embeddings, the F1-score grows to 72.3%.

For NER, the PL-Marker model uses a similar approach. For each possible span in the input starting at token v_i and ending at token $v_{j,j\geq i}$, leviated markers are created, which do not affect the embeddings of the normal tokens. Again the embeddings of the start and end tokens of a span as well as the embeddings of leviated markers are input for a logistic classifier computing the probability of the

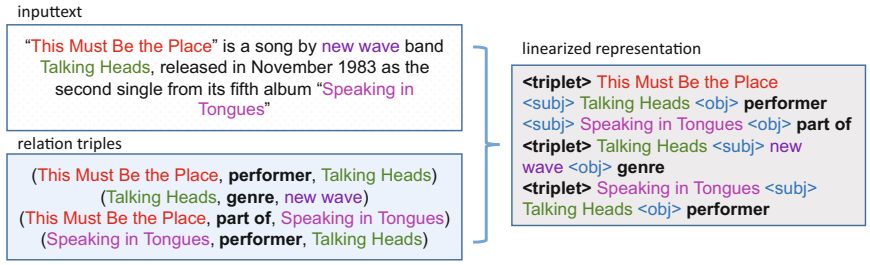

Fig. 5.6 For the training set the relation information on the left side is linearized to the representation on the right side. The REBEL model thus learns to translate the input text to this linearized representation [20]

different NE-types. The model uses an efficient 'packing' to reduce computational effort. On the CoNLL03 named entity benchmark, PL-markers with a pre-trained $\text{RoBERTa}_{\text{LARGE}}$ achieve an F1-value of 94.0, which is well below the current SOTA of 96.1% held by DeBERTa [19]. When the relation extraction employs the entity types and spans predicted by the PL-MARKER NER, the F1-value of the joint approach drops to 70.5%, which is SOTA for the ACE05 benchmark on joint NER and relation extraction.

REBEL [20] uses the encoder-decoder transformer $\text{BART}_{\text{LARGE}}$ (Sect. 3.1.3) for joint entity and relation extraction that outputs each relation (h, r, t) triplet present in the input text. It translates a raw input sentence containing entities, together with implicit relations between them, into a set of triplets that explicitly refer to those relations. An example is shown in Fig. 5.6. Each relation in the text appears in the output according to the position of its first argument. An entity may be part of different relations, which are ordered according to the position of the second argument. This defines the order of relations in the linearized representation.

The pre-trained $\text{BART}_{\text{LARGE}}$ with 400M parameters is first fine-tuned on a Wikipedia and WikiData training set with 220 relation types. Then it is fine-tuned a second time on varying benchmark datasets. On the *DocRED benchmark* [121] it achieves SOTA with an F-value of 47.1%. On the *New York Times dataset* it has a SOTA performance with 93.4% F1. On the ReTACRED benchmark it yields 90.4% F1 without the inclusion of entity type markers used by other approaches.

Aspect-Based Sentiment Analysis

Aspect-based sentiment analysis, also known as aspect-level sentiment analysis, feature-based sentiment analysis, or simply, aspect sentiment analysis, allows organizations to perform a detailed analysis of their member or customer feedback data. This ranges from analyzing customer reactions for a restaurant to evaluating the attitude to political statements made by a politician. An example is *"The waiter$_{1\text{-aspect}}$ was very friendly$_{1\text{-positive}}$, but the steak mignon$_{2\text{-aspect}}$ was*

extremely burnt$_{2\text{-}negative}$." Note that a sentence may contain different aspects and each sentiment has to be assigned to one aspect. A recent survey of aspect-based sentiment analysis is given by Zhang et al. [129].

DeBERTa (Sect. 3.1.1) is a powerful BERT-like model, which assumes that the aspects are already known. It employs a disentangled attention mechanism for computing separate attention scores between words and positions disentangling semantic (content) and syntactic (position) representation of the textual data. The objective is to determine the sentiment of each aspect of a given entity. The input consist of a text and an aspect, e.g. $x =$*"[CLS] ... nice video camera and keyboard ... [SEP] keyboard [SEP]"*, where *"keyboard"* is a possible aspect span from the text [94]. The output embedding of *[CLS]* is used as input to a logistic classifier which generates the probabilities of three possible labels positive, negative, neutral. The model is fine-tuned on the *SemEval 2014 Task 4.2 benchmark*. It yields a mean accuracy for the Restaurant and Laptop data of 86.1%. There are much more complex approaches like **LSA** (local sentiment aggregation) [119] achieving a SOTA of 88.6% on this benchmark.

GRACE [54] aims at extracting aspects and labels simultaneously. It consists of a first $\text{BERT}_{\text{BASE}}$ module generating token embeddings of the input text, which are fine-tuned to mark aspects by IOB2 tags for each token. The resulting information is fed into a Transformer decoder to predict the sentiments (positive, negative, neural) for each token. This decoder uses a multi-head cross attention to include the information from the first aspect module. Again for each token embedding in the last layer a logistic classifier is used to compute the probabilities of sentiments. To make the model more robust, small perturbations for input token embeddings are used during training. Note that no masked cross-attention is necessary as the decoder is not autoregressive. In this way, the model is able to take into account the interactions between aspect terms when labeling sentiments. The model achieves 87.9% F1 score for aspect extraction for the laptop reviews from SemEval 2014 and a SOTA of 70.7% F1-value for the joint extraction of aspects and sentiments. On the restaurant reviews it yields an F1 of 78.1% and on a tweet benchmark 58.3% for joint sentiment extraction, again outperforming a number of other models.

Semantic Role Labeling

Semantic role labeling considers a predicate (e.g. verb) of a sentence and word phrases are classified according to their syntactic roles, such as agent, goal, or result. It can be used to determine the meaning of the sentence. As an example consider the sentence *"They want to do more ."* where *"want"* is the predicate, *"They"* is the agent and *"to do more"* is the object (thing wanted).

Crf2o [133] is a tree-structured conditional random field (treecrf) [28] using contextual embeddings of the input tokens computed by RoBERTa as input. The sequence $\boldsymbol{x} = (x_1, \ldots, x_T)$ of inputs can be arranged in a tree $\boldsymbol{y}$ and gets a score, which is the sum of all scores of its subtrees $s(\boldsymbol{x}, \boldsymbol{y}) = \sum_{t \in \boldsymbol{y}} s(\boldsymbol{x}, t)$. Similar to dependency parsing, this can be used to model the dependency of phrases from the

predicate in semantic role labeling [87]. To generate all possible subtrees requires T^3 operations, which is very inefficient. The authors were able to reduce this effort using structural constraints. In addition, they could take into account the dependency between two branches of the tree, which generated a second order tree. During training the models maximize the probability of the provided tree structure of the training data for an input. *CoNLL05* [21] and *OntoNotes* [84] are two widely used benchmarks for semantic role labeling. For CoNLL05 the Crf2o yields an F1-value of 89.6% and for OntoNotes it achieves an F1-value of 88.3%, which both constitute a new SOTA. Note that this technique may also be used for *dependency parsing* [132], which describes the syntactic structure of a sentence by a tree structure.

Extracting Knowledge Graphs from Pre-trained PLMs

A systematic way to extract knowledge from big language models has been demonstrated by Wang et al. [105]. Their **MaMa** approach consist of a match stage and a map stage. The match stage generates a set of candidate facts from the text collection exploiting the internal knowledge of a language model. Similar to TransE (Sect. 3.4.1) each fact is represented as a relation triple (head, relation, tail), or (h, r, t). A language model is used to generate tokens corresponding to r or t. As a condition, the r values should be contiguous text sequences and express frequent relations.

In the map stage the triples are mapped to related triples with appropriate relations. As an example (Dylan, is, songwriter) is mapped to (Bob Dylan.Q392, occupation.P106, Songwriter.Q753110) according to the Wikidata schema. This stage is related to entity linking discussed in Sect. 5.3.3. The reason for mapping to an existing KG schema is to make use of the high-quality schema designed by experts.

A subgraph of the generated relations is shown in Fig. 5.7. Compared to the SOTA information extraction system Stanford OpenIE [5] with 27.1% F1-value the approach yields 29.7% F1-value. The authors report that performance increases with model size because larger models can store more knowledge.

Available Implementations

- PL-Marker Code and models are publicly available at https://github.com/thunlp/PL-Marker.
- REBEL on GitHub https://github.com/babelscape/rebel and Hugging Face https://huggingface.co/Babelscape/rebel-large
- MaMa: Source code and pre-trained models at https://github.com/theblackcat102/language-models-are-knowledge-graphs-pytorch

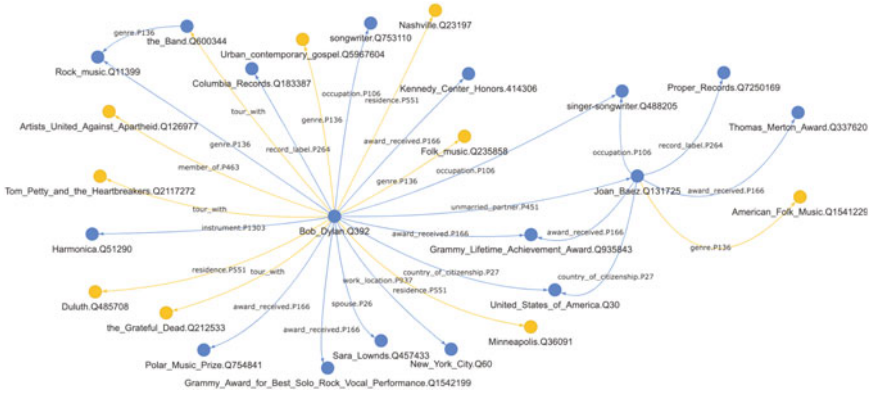

Fig. 5.7 A snapshot subgraph of the open KG generated by MAMA [105] using $\text{BERT}_{\text{LARGE}}$ from Wikipedia pages neighboring *"Bob Dylan"*. The blue node and arrow represent the mapped facts in the Wikidata schema, while the yellow node and arrow denote the unmapped facts in the open schema. The correct facts that are new in Wikidata are visualized in yellow. Image source: [105, p. 6], with kind permission of the authors

5.4.5 Distant Supervision

Obtaining a large annotated dataset for relation extraction is a tedious task and often difficult due to privacy issues. Since much relational knowledge is stored in knowledge bases, Mintz et al. [65] proposed the *distant supervision* paradigm. The idea behind it is to collect all text mentions where two entities co-occur, which are in a relation in the knowledge base. Then it is assumed that for this mention pair the relation holds. Since this is not correct for all such mention pairs, many approaches aim to combat this 'noise'. One approach is *multi-instance learning*, which relaxes the original assumption that all text mention pairs represent the relation to the assumption that the relation holds for at least one pair [2, 137], or a specified fraction like 10% or depending on a score value. Take for example the entities *"Barack Obama"* and *"Hawaii"*, which might be in a relation *"born_in"* in a KB. Sentences obtained by searching for occurrences of these two entities could be *"Obama was born in Hawaii"* as well as *"Obama was on family vacation in Hawaii"*, where only the former represents the relation and should be used for training.

KGPool [67] uses entity pairs obtained from a KB, but also attributes associated with them. The idea is to create representations of the entity nodes, the sentence in which they occur, and the attributes of the entity nodes in a knowledge base, such as their description, instance-of and alias attribute. All this information is embedded using word and character embeddings and bidirectional LSTMs and connected as a heterogeneous information graph. Next three layers of graph convolutional networks are used with readout layers. Only relevant attribute nodes are picked by using self-attention on the readout representations, calculating a softmax score and then filtering via a hyperparameter according to the scores. A dynamic mask

is created which pools out the less essential entity attribute nodes. Finally, all intermediate representations of both entities, the sentence and the readouts are each concatenated to form the final entity, sentence and readout representation. These representations together with relation representations are then passed through a fully connected layer with softmax activation to calculate the scores per relation. The *New York Times dataset* is a standard benchmark for relation extraction with distant supervision. KGPool achieves a SOTA precision@10 of 92.3%, which is the fraction of relevant results if the 'best' 10 of the matches are used.

5.4.6 Relation Extraction Using Layout Information

To understand a formal text, often the document layout has be taken into account in addition to its text. Especially in form-like texts, the positions of words and filled-in values are important. In Sect. 7.2 we will describe, how text and images can be simultaneously processed by one or more transformers to extract meaning from both media. In anticipation, we will use this ability of transformers to process multimodal inputs and additionally include layout information via 2-dimensional positional features. A comprehensive overview of progress in layout analysis is provided by Stanisławek [96]. We will focus on methods for key-value extraction in this subchapter. In the task of key-value extraction, documents are analyzed to extract printed values to written keys of interest. Sample applications are the automatic processing of invoices, in which keys are attributes such as invoice date or the total amount to be paid.

ReLIE [57] is a framework for key-value extraction from form-like documents. The candidate generation step has the purpose of finding all possible value candidates for a certain key, e.g. the value *"1/16/2018"* for the key *"Date"*. Often these value candidates correspond to basic types such as numbers, amounts, dates, etc. and can be found via rule based matchers. Then a transformer-based scoring model is trained, to identify valid values among the extracted value candidates. To this end, embeddings are learned for the keys, the position of the value candidate and for neighboring tokens and their positions. Positions of a value candidate and each of its neighbors are described using the 2-D Cartesian coordinates of the centroids of their respective bounding boxes. Note that the text of the candidate value is not encoded to avoid overfitting. All embeddings are related to each other by self-attention in an autoencoder. The field embedding and the candidate embedding are then compared via cosine similarity and the resulting score is scaled into a range of [0, 1]. The model achieves an f1-score of 87.8% on key-value extraction for invoices and 83.3% for receipts.

DocFormer [6] consists of a CNN visual backbone and an encoder-only transformer architecture. Visual embeddings of the document are produced via a ResNet50 model and projected to the appropriate embedding size via a linear layer. Text tokens are contained in a bounding box and the top-left and lower-right position of each token bounding box are transformed to embeddings by two

different matrices. In addition, the height, width and distances between neighboring bounding boxes are encoded. The 2D-positional embeddings are enriched with absolute positions via 1D-positional embeddings. Separate spatial embeddings are trained for visual and textual features. The attention mechanism of the DocFormer is a modified version of the original attention mechanism. Separate attention scores are calculated for the visual and the textual representation of tokens. In addition to the key-query attention, the relative position embeddings of both query and key tokens are used to add relative position attentions as well as a spatial attention for both the visual and the textual embeddings. The spatial attention weights are shared between the visual and the textual representations.

DocFormer is pre-trained with three different pre-training tasks: multi-modal masked language modeling (MM-MLM), learn to reconstruct (LTR) and text describes image (TDI). In the MM-MLM task, tokens are masked and should be reconstructed by the model. In LTR, the model is tasked to reconstruct the image of a document, given the multi-modal representation. A smooth-L1 loss is used to calculate differences between the original and the reconstructed image. TDI requires a text-image matching task, in which the model has to predict for random samples whether the image and the text are aligned or not. The *FUNSD benchmark* [38] considers forms in 199 scanned documents, where tokens have to be assigned to a semantic key, such as 'question' or 'answer'. On FUNSD DocFormer reaches an F1-value of 84.6%, which was SOTA at publication time.

LayoutLM3 [34] uses an image embedding method inspired by the Vision Transformer (Sect. 7.2.2). Each image is partitioned into 16×16 image patches similar to the Vision Transformer and linearly transformed to embeddings. As shown in Fig. 5.8 words and image patches are processed by the same autoregressive Transformer. For pre-training the model uses the masked language modeling task, masked image patches and word-patch alignment pre-training task. In the masked image patches task, image patches have to be reconstructed by the model. The word-patch alignment task has to enable the model to learn alignments between textual and visual representations. The model should classify whether text and image patch of a token are aligned, i.e. both are unmasked, or unaligned, i.e. the image patch is masked. The *PubLayNet benchmark* [134] contains the document layout of more than 1 million pdf documents matched against the correct document structure. Here LayoutLM3 achieves SOTA with 94.5% mean average precision of bounding boxes. It outperforms DocFormer on the FUNSD key-value extraction tasks and other benchmarks. *LayoutXLM* is a recent multilingual version of LayoutLM2 [116].

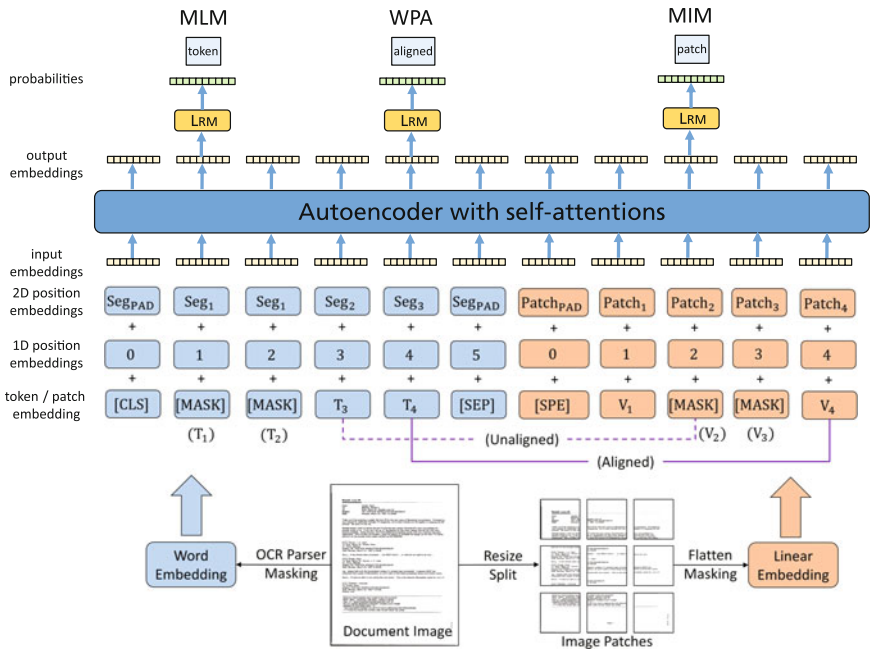

Fig. 5.8 LayoutLMv3 takes the linear projection of image patches and word tokens as inputs and encodes them into contextualized vector representations. LayoutLMv3 is pre-trained with discrete token reconstructive objectives of Masked Language Modeling (MLM) and Masked Image Modeling (MIM). Additionally, LayoutLMv3 is pre-trained with a Word-Patch Alignment (WPA) objective to learn cross-modal alignment by predicting whether the corresponding image patch of a text word is masked. "Seg" denotes segment-level positions. Image source: [34, p. 3], printed with kind permission of the authors

Available Implementations

- KGPool at https://github.com/nadgeri14/KGPool

5.4.7 Summary

Relation extraction has the task to evaluate the expressed relationship in the text with respect to specific entities. An example is the assessment of certain product characteristics by customers, which can help to improve the product or service. Given the massive amount of textual content, it is intractable to manually process the opinion information.

For simple cases, the relation arguments are know and relation extraction can be solved as a simple classification task using some BERT variant like RoBERTa,

DeBERTa, or SpanBERT. However, to actually use these models we have to extract the relation arguments in a prior step, which leads to an increased total error.

More challenging is the simultaneous extraction of relation arguments and the corresponding relation type, as these task depend on each other. UniRE annotates entities and relations in a joint matrix and introduces a corresponding bias into the self-attention computations. PL-marker marks the first relation arguments with special tokens and the second argument with so-called leviated tokens. These tokens have specific attention properties and are able to improve the performance on popular benchmarks. GRACE employs a specific encoder-decoder architecture where the encoder labels the relation arguments (aspects) and the decoder assigns relation tags to each token. REBEL uses the BART encoder-decoder to translate the input sentence to a unique representation of the covered relations.

Relation extraction models have been adapted to specific applications. GRACE has been tuned for aspect-based sentiment analysis and Crf2o to semantic role labeling. The latter uses contextual embeddings and determines the relation between predicate and corresponding phrases by an efficient TreeCRF. Finally, MaMa can be used to build a knowledge graph from extracted relations between entities.

Often the spatial layout of documents and web pages contains relevant information for the extraction of relation arguments. In this case, visual information from the document image can be exploited to arrive at a valid interpretation. This visual information can be included via the position of bounding boxes for keys and values, but also in the form of image patches, which are explored later with the image transformer.

All recent relation extraction approaches are based on PLMs. Most models use small BERT variants for their experiments. Therefore, it can be assumed that larger models will directly increase performance. In addition, Foundation Models like GPT-3 may be fine-tuned (Sect. 3.6.2) and probably will result in a higher accuracy. A related alternative is InstructGPT (Sect. 3.6.5), which can be easily directed to perform a relation extraction via question answering, e.g. *"Who built the statue of liberty?"* [77, p. 29]. However, it seems to be difficult to evaluate the performance of this approach with respect to some test data.

References

1. J. Abreu, L. Fred, D. Macêdo, and C. Zanchettin. "Hierarchical Attentional Hybrid Neural Networks for Document Classification". In: *Int. Conf. Artif. Neural Netw*. Springer, 2019, pp. 396–402.
2. L. Adilova, S. Giesselbach, and S. Rüping. "Making Efficient Use of a Domain Expert's Time in Relation Extraction". 2018. arXiv: 1807.04687.
3. N. Alex et al. "RAFT: A Real-World Few-Shot Text Classification Benchmark". Jan. 18, 2022. arXiv: 2109.14076 [cs].
4. Z. Alyafeai, M. S. AlShaibani, and I. Ahmad. "A Survey on Transfer Learning in Natural Language Processing". 2020. arXiv: 2007.04239.

5. G. Angeli, M. J. J. Premkumar, and C. D. Manning. "Leveraging Linguistic Structure for Open Domain Information Extraction". In: *Proc. 53rd Annu. Meet. Assoc. Comput. Linguist. 7th Int. Jt. Conf. Nat. Lang. Process. Vol. 1 Long Pap*. 2015, pp. 344–354.
6. S. Appalaraju, B. Jasani, B. U. Kota, Y. Xie, and R. Manmatha. "Docformer: End-to-end Transformer for Document Understanding". In: *Proc. IEEECVF Int. Conf. Comput. Vis*. 2021, pp. 993–1003.
7. S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives. "DBpedia: A Nucleus for a Web of Open Data". In: *Semantic Web*. Ed. by K. Aberer et al. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2007, pp. 722–735. ISBN: 978-3-540-76298-0. https://doi.org/10.1007/978-3-540-76298-0_52.
8. E. Barba, T. Pasini, and R. Navigli. "ESC: Redesigning WSD with Extractive Sense Comprehension". In: *Proc. 2021 Conf. North Am. Chapter Assoc. Comput. Linguist. Hum. Lang. Technol*. 2021, pp. 4661–4672.
9. E. Barba, L. Procopio, N. Campolungo, T. Pasini, and R. Navigli. "MuLaN: Multilingual Label propagatioN for Word Sense Disambiguation". In: *Proc IJCAI*. 2020, pp. 3837–3844.
10. E. Barba, L. Procopio, and R. Navigli. "ConSeC: Word Sense Disambiguation as Continuous Sense Comprehension". In: *Proc. 2021 Conf. Empir. Methods Nat. Lang. Process*. 2021, pp. 1492–1503.
11. I. Beltagy, K. Lo, and A. Cohan. "SciBERT: A Pretrained Language Model for Scientific Text". 2019. arXiv: 1903.10676.
12. M. Bevilacqua and R. Navigli. "Breaking through the 80% Glass Ceiling: Raising the State of the Art in Word Sense Disambiguation by Incorporating Knowledge Graph Information". In: *Proc Assoc. Comput. Linguist*. 2020, pp. 2854–2864.
13. M. Bevilacqua, T. Pasini, A. Raganato, and R. Navigli. "Recent Trends in Word Sense Disambiguation: A Survey". In: *Proc. Thirtieth Int. Jt. Conf. Artif. Intell. IJCAI-21*. International Joint Conference on Artificial Intelligence, Inc, 2021.
14. K. Bhatia, K. Dahiya, H. Jain, P. Kar, A. Mittal, Y. Prabhu, and M. Varma. *The Extreme Classification Repository*. June 7, 2021. URL: http://manikvarma.org/downloads/XC/XMLRepository.html (visited on 06/07/2021).
15. T. Blevins and L. Zettlemoyer. "Moving down the Long Tail of Word Sense Disambiguation with Gloss-Informed Biencoders". 2020. arXiv: 2005.02590.
16. P. Bojanowski. fastText. 2016. URL: https://fasttext.cc/index.html (visited on 02/21/2021).
17. F. Bond and R. Foster. "Linking and Extending an Open Multilingual Wordnet". In: *Proc. 51st Annu. Meet. Assoc. Comput. Linguist. Vol. 1 Long Pap*. 2013, pp. 1352–1362.
18. P. Bose, S. Srinivasan, W. C. Sleeman, J. Palta, R. Kapoor, and P. Ghosh. "A Survey on Recent Named Entity Recognition and Relationship Extraction Techniques on Clinical Texts". In: *Appl. Sci*. 11.18 (2021), p. 8319.
19. F. Brandon. Brandon25/Deberta-Base-Finetuned-Ner · Hugging Face. Oct. 12, 2021. URL: https://huggingface.co/brandon25/deberta-base-finetuned-ner (visited on 02/15/2022).
20. P.-L. H. Cabot and R. Navigli. "REBEL: Relation Extraction By End-to-end Language Generation". In: *Find. Assoc. Comput. Linguist. EMNLP 2021*. 2021, pp. 2370–2381.
21. X. Carreras and L. Màrquez. "Introduction to the CoNLL-2005 Shared Task: Semantic Role Labeling". In: *Proc. Ninth Conf. Comput. Nat. Lang. Learn. CoNLL-2005*. 2005, pp. 152–164.
22. W.-C. Chang et al. "Extreme Multi-label Learning for Semantic Matching in Product Search". June 23, 2021. arXiv: 2106.12657 [cs].
23. G. Choi, S. Oh, and H. Kim. "Improving Document-Level Sentiment Classification Using Importance of Sentences". In: *Entropy* 22.12 (2020), p. 1336.
24. A. D. Cohen, S. Rosenman, and Y. Goldberg. "Relation Extraction as Two-way Span-Prediction". 2020. arXiv: 2010.04829.
25. N. De Cao, G. Izacard, S. Riedel, and F. Petroni. "Autoregressive Entity Retrieval". Mar. 24, 2021. arXiv: 2010.00904.
26. S. Ding, J. Shang, S. Wang, Y. Sun, H. Tian, H. Wu, and H. Wang. "ERNIE-DOC: The Retrospective Long-Document Modeling Transformer". 2020. arXiv: 2012.15688.
27. V. Dobrovolskii. "Word-Level Coreference Resolution". 2021. arXiv: 2109.04127.

28. J. Eisner. “Bilexical Grammars and Their Cubic-Time Parsing Algorithms”. In: *Advances in Probabilistic and Other Parsing Technologies*. Springer, 2000, pp. 29–61.
29. D. Gillick, S. Kulkarni, L. Lansing, A. Presta, J. Baldridge, E. Ie, and D. Garcia-Olano. “Learning Dense Representations for Entity Retrieval”. 2019. arXiv: 1909.10506.
30. GitHub. *GitHub*. 2021. URL: https://github.com/.
31. Gu. *BLURB Leaderboard*. 2021. URL: https://microsoft.github.io/BLURB/ (visited on 02/13/2022).
32. J. He, L. Wang, L. Liu, J. Feng, and H. Wu. “Long Document Classification from Local Word Glimpses via Recurrent Attention Learning”. In: *IEEE Access* 7 (2019), pp. 40707–40718.
33. L. Huang, C. Sun, X. Qiu, and X. Huang. “GlossBERT: BERT for Word Sense Disambiguation with Gloss Knowledge”. 2019. arXiv: 1908.07245.
34. Y. Huang, T. Lv, L. Cui, Y. Lu, and F. Wei. “LayoutLMv3: Pre-training for Document AI with Unified Text and Image Masking”. 2022. arXiv: 2204.08387.
35. huggingface. *Transformers – Transformers 4.3.0 Documentation*. 2021. URL: https://huggingface.co/transformers/ (visited on 02/21/2021).
36. M. S. Jahan and M. Oussalah. “A Systematic Review of Hate Speech Automatic Detection Using Natural Language Processing”. 2021. arXiv: 2106.00742.
37. K. Jasinska, K. Dembczynski, R. Busa-Fekete, K. Pfannschmidt, T. Klerx, and E. Hullermeier. “Extreme F-Measure Maximization Using Sparse Probability Estimates”. In: *Int. Conf. Mach. Learn*. PMLR, 2016, pp. 1435–1444.
38. G. Jaume, H. K. Ekenel, and J.-P. Thiran. “Funsd: A Dataset for Form Understanding in Noisy Scanned Documents”. In: *2019 Int. Conf. Doc. Anal. Recognit. Workshop ICDARW*. Vol. 2. IEEE, 2019, pp. 1–6.
39. T. Jiang, D. Wang, L. Sun, H. Yang, Z. Zhao, and F. Zhuang. “Lightxml: Transformer with Dynamic Negative Sampling for High-Performance Extreme Multi-Label Text Classification”. 2021. arXiv: 2101.03305.
40. J. Johnson, M. Douze, and H. Jégou. “Billion-Scale Similarity Search with Gpus”. In: *IEEE Trans. Big Data* (2019).
41. M. Joshi, D. Chen, Y. Liu, D. S. Weld, L. Zettlemoyer, and O. Levy. “Spanbert: Improving Pre-Training by Representing and Predicting Spans”. In: *Trans. Assoc. Comput. Linguist*. 8 (2020), pp. 64–77.
42. K. raj Kanakarajan, B. Kundumani, and M. Sankarasubbu. “BioELECTRA:Pretrained Biomedical Text Encoder Using Discriminators”. In: *Proc. 20th Workshop Biomed. Lang. Process*. BioNLP-NAACL 2021. Online: Association for Computational Linguistics, June 2021, pp. 143–154. https://doi.org/10.18653/v1/2021.bionlp-1.16.
43. J.-D. Kim, T. Ohta, Y. Tateisi, and J. Tsujii. “GENIA Corpus-a Semantically Annotated Corpus for Bio-Textmining”. In: *Bioinformatics* 19 (suppl_1 2003), pp. i180–i182.
44. K. Kowsari, K. Jafari Meimandi, M. Heidarysafa, S. Mendu, L. Barnes, and D. Brown. “Text Classification Algorithms: A Survey”. In: *Information* 10.4 (2019), p. 150.
45. Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut. “Albert: A Lite BERT for Self-Supervised Learning of Language Representations”. 2020. arXiv: 1909.11942.
46. H. Langone, B. R. Haskell, and G. A. Miller. Annotating Wordnet. PRINCETON UNIV NJ COGNITIVE SCIENCE LAB, 2004.
47. Q. V. Le and T. Mikolov. “Distributed Representations of Sentences and Documents”. May 22, 2014. arXiv: 1405.4053 [cs].
48. J. Li, A. Sun, J. Han, and C. Li. “A Survey on Deep Learning for Named Entity Recognition”. In: *IEEE Trans. Knowl. Data Eng*. (2020).
49. Q. Li et al. “A Survey on Text Classification: From Shallow to Deep Learning”. 2020. arXiv: 2008.00364.
50. X. Li, J. Feng, Y. Meng, Q. Han, F. Wu, and J. Li. “A Unified MRC Framework for Named Entity Recognition”. 2019. arXiv: 1910.11476.
51. X. Liu, W.-C. Chang, H.-F. Yu, C.-J. Hsieh, and I. S. Dhillon. “Label Disentanglement in Partition-based Extreme Multilabel Classification”. 2021. arXiv: 2106.12751.

52. D. Loureiro, K. Rezaee, M. T. Pilehvar, and J. Camacho-Collados. “Analysis and Evaluation of Language Models for Word Sense Disambiguation”. In: *Comput. Linguist. 2021 47 2 387–443* (Mar. 17, 2021).
53. E. Loza Mencía and J. Fürnkranz. “Efficient Pairwise Multilabel Classification for Large-Scale Problems in the Legal Domain”. In: *Jt. Eur. Conf. Mach. Learn. Knowl. Discov. Databases*. Springer, 2008, pp. 50–65.
54. H. Luo, L. Ji, T. Li, N. Duan, and D. Jiang. “Grace: Gradient Harmonized and Cascaded Labeling for Aspect-Based Sentiment Analysis”. 2020. arXiv: 2009.10557.
55. S. Lyu and H. Chen. “Relation Classification with Entity Type Restriction”. 2021. arXiv: 2105.08393.
56. A. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts. “Learning Word Vectors for Sentiment Analysis”. In: *Proc. 49th Annu. Meet. Assoc. Comput. Linguist. Hum. Lang. Technol*. 2011, pp. 142–150.
57. B. P. Majumder, N. Potti, S. Tata, J. B. Wendt, Q. Zhao, and M. Najork. “Representation Learning for Information Extraction from Form-like Documents”. In: *Proc. 58th Annu. Meet. Assoc. Comput. Linguist*. 2020, pp. 6495–6504.
58. T. Mandl, S. Modha, P. Majumder, D. Patel, M. Dave, C. Mandlia, and A. Patel. “Overview of the HASOC Track at FIRE 2019: Hate Speech and Offensive Content Identification in Indo-European Languages”. In: *Proc. 11th Forum Inf. Retr. Eval*. FIRE ’19: Forum for Information Retrieval Evaluation. Kolkata India: ACM, Dec. 12, 2019, pp. 14–17. ISBN: 978-1-4503-7750-8. https://doi.org/10.1145/3368567.3368584.
59. B. Mathew, P. Saha, S. M. Yimam, C. Biemann, P. Goyal, and A. Mukherjee. “HateXplain: A Benchmark Dataset for Explainable Hate Speech Detection”. 2021. arXiv: 2012.10289 [cs].
60. J. McAuley and J. Leskovec. “Hidden Factors and Hidden Topics: Understanding Rating Dimensions with Review Text”. In: *Proc. 7th ACM Conf. Recomm. Syst*. 2013, pp. 165–172.
61. R. Mihalcea. *SemCor Corpus*. June 13, 2008. URL: https://kaggle.com/nltkdata/semcorcorpus (visited on 01/04/2022).
62. G. A. Miller. “WordNet: A Lexical Database for English”. In: *Commun. ACM* 38.11 (1995), pp. 39–41.
63. G. A. Miller, C. Leacock, R. Tengi, and R. T. Bunker. “A Semantic Concordance”. In: *Hum. Lang. Technol. Proc. Workshop Held Plainsboro N. J. March 21–24 1993*. 1993.
64. S. Minaee, N. Kalchbrenner, E. Cambria, N. Nikzad, M. Chenaghlu, and J. Gao. “Deep Learning-Based Text Classification: A Comprehensive Review”. In: *ACM Comput. Surv. CSUR* 54.3 (2021), pp. 1–40.
65. M. Mintz, S. Bills, R. Snow, and D. Jurafsky. “Distant Supervision for Relation Extraction without Labeled Data”. In: *Proc. Jt. Conf. 47th Annu. Meet. ACL 4th Int. Jt. Conf. Nat. Lang. Process. AFNLP*. 2009, pp. 1003–1011.
66. A. Moro and R. Navigli. “Semeval-2015 Task 13: Multilingual All-Words Sense Disambiguation and Entity Linking”. In: *Proc. 9th Int. Workshop Semantic Eval. SemEval 2015*. 2015, pp. 288–297.
67. A. Nadgeri, A. Bastos, K. Singh, I. O. Mulang, J. Hoffart, S. Shekarpour, and V. Saraswat. “Kgpool: Dynamic Knowledge Graph Context Selection for Relation Extraction”. 2021. arXiv: 2106.00459.
68. Z. Nasar, S. W. Jaffry, and M. K. Malik. “Named Entity Recognition and Relation Extraction: State-of-the-art”. In: *ACM Comput. Surv. CSUR* 54.1 (2021), pp. 1–39.
69. R. Navigli. “Word Sense Disambiguation: A Survey”. In: *ACM Comput. Surv. CSUR* 41.2 (2009), pp. 1–69.
70. R. Navigli, D. Jurgens, and D. Vannella. “Semeval-2013 Task 12: Multilingual Word Sense Disambiguation”. In: *Second Jt. Conf. Lex. Comput. Semant. SEM Vol. 2 Proc. Seventh Int. Workshop Semantic Eval. SemEval 2013*. 2013, pp. 222–231.
71. R. Navigli and S. P. Ponzetto. “BabelNet: The Automatic Construction, Evaluation and Application of a Wide-Coverage Multilingual Semantic Network”. In: *Artif. Intell*. 193 (2012), pp. 217–250.

72. ner. *Papers with Code - Named Entity Recognition*. 2021. URL: https://paperswithcode.com/task/named-entity-recognition-ner (visited on 07/09/2021).
73. NIH. *Download Data*. PubMed. 2022. URL: https://pubmed.ncbi.nlm.nih.gov/download/ (visited on 06/15/2022).
74. NLP. *The NLP Index*. 2021. URL: https://index.quantumstat.com/.
75. Omegawiki. *OmegaWiki*. 2021. URL: http://www.omegawiki.org/ (visited on 01/03/2022).
76. OpenAi. *OpenAI API*. 2021. URL: https://beta.openai.com (visited on 11/14/2021).
77. L. Ouyang et al. "Training Language Models to Follow Instructions with Human Feedback". Jan. 31, 2022. arXiv: 2203.02155.
78. G. Paaß and F. Reichartz. "Exploiting Semantic Constraints for Estimating Supersenses with CRFs". In: *Proc. 2009 SIAM Int. Conf. Data Min.* SIAM, 2009, pp. 485–496.
79. Papers-with-code. *Papers with Code*. 2021. URL: https://paperswithcode.com/.
80. M. E. Peters, M. Neumann, R. L. Logan IV, R. Schwartz, V. Joshi, S. Singh, and N. A. Smith. "Knowledge Enhanced Contextual Word Representations". 2019. arXiv: 1909.04164.
81. M. T. Pilehvar, J. Camacho-Collados, R. Navigli, and N. Collier. "Towards a Seamless Integration of Word Senses into Downstream Nlp Applications". 2017. arXiv: 1710.06632.
82. A. Pilz and G. Paaß. "From Names to Entities Using Thematic Context Distance". In: *Proc. 20th ACM Int. Conf. Inf. Knowl. Manag.* 2011, pp. 857–866.
83. Y. Prabhu, A. Kag, S. Harsola, R. Agrawal, and M. Varma. "Parabel: Partitioned Label Trees for Extreme Classification with Application to Dynamic Search Advertising". In: *Proc. 2018 World Wide Web Conf.* 2018, pp. 993–1002.
84. S. Pradhan, A. Moschitti, N. Xue, O. Uryupina, and Y. Zhang. "CoNLL-2012 Shared Task: Modeling Multilingual Unrestricted Coreference in OntoNotes". In: *Jt. Conf. EMNLP CoNLL-Shar. Task*. 2012, pp. 1–40.
85. J. W. Rae et al. "Scaling Language Models: Methods, Analysis & Insights from Training Gopher". In: *ArXiv Prepr. ArXiv211211446* (Dec. 8, 2021), p. 118.
86. P. Ramachandran, B. Zoph, and Q. V. Le. "Searching for Activation Functions". 2017. arXiv: 1710.05941.
87. F. Reichartz, H. Korte, and G. Paass. "Semantic Relation Extraction with Kernels over Typed Dependency Trees". In: *Proc. 16th ACM SIGKDD Int. Conf. Knowl. Discov. Data Min.* 2010, pp. 773–782.
88. S. G. Roy, U. Narayan, T. Raha, Z. Abid, and V. Varma. "Leveraging Multilingual Transformers for Hate Speech Detection". 2021. arXiv: 2101.03207.
89. E. F. Sang and F. De Meulder. "Introduction to the CoNLL-2003 Shared Task: Language-independent Named Entity Recognition". 2003. arXiv: cs/0306050.
90. T. Schick and H. Schütze. "True Few-Shot Learning with Prompts – A Real-World Perspective". Nov. 26, 2021. arXiv: 2111.13440 [cs].
91. P. Schmid. *Few-Shot Learning in Practice: GPT-Neo and the .. Accelerated Inference API*. June 3, 2021. URL: https://huggingface.co/blog/few-shot-learning-gpt-neo-and-inference-api (visited on 05/23/2022).
92. O. Sevgili, A. Shelmanov, M. Arkhipov, A. Panchenko, and C. Biemann. "Neural Entity Linking: A Survey of Models Based on Deep Learning". 2020. arXiv: 2006.00575.
93. Y. Shen, X. Ma, Z. Tan, S. Zhang, W. Wang, and W. Lu. "Locate and Label: A Two-stage Identifier for Nested Named Entity Recognition". 2021. arXiv: 2105.06804.
94. E. H. Silva and R. M. Marcacini. "Aspect-Based Sentiment Analysis Using BERT with Disentangled Attention". In: (2021). URL: https://repositorio.usp.br/bitstreams/701d2a63-e3f4-450d-8617-ad80de4345ed.2185FoundationModelsforInformationExtraction
95. Spanbert. *Papers with Code - The Latest in Machine Learning*. July 17, 2021. URL: https://paperswithcode.com/paper/spanbert-improving-pre-training-by/review/?hl=28781 (visited on 07/17/2021).
96. T. Stanisławek. *Awesome Document Understanding*. July 2, 2022. URL: https://github.com/tstanislawek/awesome-document-understanding (visited on 07/08/2022).
97. G. Stoica, E. A. Platanios, and B. Póczos. "Re-Tacred: Addressing Shortcomings of the Tacred Dataset". In: *Proc. AAAI Conf. Artif. Intell.* Vol. 35. 15. 2021, pp. 13843–13850.

98. N. Stylianou and I. Vlahavas. “A Neural Entity Coreference Resolution Review”. In: *Expert Syst. Appl*. 168 (2021), p. 114466.
99. Y. Sun et al. “Ernie: Enhanced Representation through Knowledge Integration”. 2019. arXiv: 1904.09223.
100. C. Sutton and A. McCallum. “An Introduction to Conditional Random Fields for Relational Learning”. In: *Introd. Stat. Relational Learn*. 2 (2006), pp. 93–128.
101. T. Thongtan and T. Phienthrakul. “Sentiment Classification Using Document Embeddings Trained with Cosine Similarity”. In: *Proc. 57th Annu. Meet. Assoc. Comput. Linguist. Stud. Res. Workshop*. Florence, Italy: Association for Computational Linguistics, July 2019, pp. 407–414. https://doi.org/10.18653/v1/P19-2057.
102. R. Tinn et al. “Fine-Tuning Large Neural Language Models for Biomedical Natural Language Processing”. Dec. 14, 2021. arXiv: 2112.07869 [cs].
103. S. Toshniwal, S. Wiseman, A. Ettinger, K. Livescu, and K. Gimpel. “Learning to Ignore: Long Document Coreference with Bounded Memory Neural Networks”. 2020. arXiv: 2010.02807.
104. C. Walker, S. Strassel, J. Medero, and K. Maeda. *ACE 2005 Multilingual Training Corpus*. Linguistic Data Consortium, Feb. 15, 2006. https://doi.org/10.35111/MWXC-VH88.
105. C. Wang, X. Liu, and D. Song. “Language Models Are Open Knowledge Graphs”. Oct. 22, 2020. arXiv: 2010.11967.
106. X. Wang, Y. Jiang, N. Bach, T. Wang, Z. Huang, F. Huang, and K. Tu. “Automated Concatenation of Embeddings for Structured Prediction”. 2020. arXiv: 2010.05006.
107. X. Wang, Y. Jiang, N. Bach, T. Wang, Z. Huang, F. Huang, and K. Tu. “Improving Named Entity Recognition by External Context Retrieving and Cooperative Learning”. 2021. arXiv: 2105.03654.
108. Y. Wang, C. Sun, Y. Wu, H. Zhou, L. Li, and J. Yan. “UniRE: A Unified Label Space for Entity Relation Extraction”. 2021. arXiv: 2107.04292.
109. R. Weischedel, M. Palmer, R. B. S. P. L. Ramshaw, N. Xue, and E. Hovy. “Ontonotes: A Large Training Corpus for Enhanced Processing”. In: *Joseph Olive Caitlin Christ. And- John McCary Ed. Handb. Nat. Lang. Mach. Transl. DARPA Glob. Lang. Exploit*. (2011).
110. G. Wiedemann, S. M. Yimam, and C. Biemann. “UHH-LT at SemEval-2020 Task 12: Fine-Tuning of Pre-Trained Transformer Networks for Offensive Language Detection”. June 10, 2020. arXiv: 2004.11493 [cs].
111. wiktionary. *Wiktionary*. 2021. URL: https://www.wiktionary.org/ (visited on 01/03/2022).
112. R. J. Williams. “Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning”. In: *Mach. Learn*. 8.3 (1992), pp. 229–256.
113. L. Wu, F. Petroni, M. Josifoski, S. Riedel, and L. Zettlemoyer. “Scalable Zero-shot Entity Linking with Dense Entity Retrieval”. In: *Proc. 2020 Conf. Empir. Methods Nat. Lang. Process. EMNLP*. 2020, pp. 6397–6407.
114. W. Wu, F. Wang, A. Yuan, F. Wu, and J. Li. “Coreference Resolution as Query-Based Span Prediction”. July 18, 2020. arXiv: 1911.01746.
115. B. Xu, Q. Wang, Y. Lyu, Y. Zhu, and Z. Mao. “Entity Structure Within and Throughout: Modeling Mention Dependencies for Document-Level Relation Extraction”. 2021. arXiv: 2102.10249.
116. Y. Xu et al. “Layoutxlm: Multimodal Pre-Training for Multilingual Visually-Rich Document Understanding”. 2021. arXiv: 2104.08836.
117. I. Yamada, A. Asai, H. Shindo, H. Takeda, and Y. Matsumoto. “LUKE: Deep Contextualized Entity Representations with Entity-Aware Self-Attention”. 2020. arXiv: 2010.01057.
118. I. Yamada, K. Washio, H. Shindo, and Y. Matsumoto. “Global Entity Disambiguation with Pretrained Contextualized Embeddings of Words and Entities”. Nov. 24, 2021. arXiv: 1909.00426 [cs].
119. H. Yang, B. Zeng, M. Xu, and T. Wang. “Back to Reality: Leveraging Pattern-driven Modeling to Enable Affordable Sentiment Dependency Learning”. 2021. arXiv: 2110.08604.
120. Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. R. Salakhutdinov, and Q. V. Le. “Xlnet: Generalized Autoregressive Pretraining for Language Understanding”. In: *Adv. Neural Inf. Process. Syst*. 2019, pp. 5753–5763.

121. Y. Yao et al. “DocRED: A Large-Scale Document-Level Relation Extraction Dataset”. 2019. arXiv: 1906.06127.
122. D. Ye, Y. Lin, and M. Sun. “Pack Together: Entity and Relation Extraction with Levitated Marker”. 2021. arXiv: 2109.06067.
123. W. Yin and A. Zubiaga. “Towards Generalisable Hate Speech Detection: A Review on Obstacles and Solutions”. In: *PeerJ Comput. Sci.* 7 (2021), e598.
124. R. You, Z. Zhang, Z. Wang, S. Dai, H. Mamitsuka, and S. Zhu. “Attentionxml: Label Tree-Based Attention-Aware Deep Model for High-Performance Extreme Multi-Label Text Classification”. 2018. arXiv: 1811.01727.
125. J. Yu, B. Bohnet, and M. Poesio. “Named Entity Recognition as Dependency Parsing”. 2020. arXiv: 2005.07150.
126. Z. Yuan, Y. Liu, C. Tan, S. Huang, and F. Huang. “Improving Biomedical Pretrained Language Models with Knowledge”. 2021. arXiv: 2104.10344.
127. M. Zaheer et al. “Big Bird: Transformers for Longer Sequences”. In: *Adv. Neural Inf. Process. Syst.* 33 (Jan. 8, 2021).
128. M. Zampieri et al. “SemEval-2020 Task 12: Multilingual Offensive Language Identification in Social Media (OffensEval 2020)”. 2020. arXiv: 2006.07235.
129. W. Zhang, X. Li, Y. Deng, L. Bing, and W. Lam. *A Survey on Aspect-Based Sentiment Analysis: Tasks, Methods, and Challenges*. Mar. 2, 2022. https://doi.org/10.48550/2203.01054. arXiv: 2203.01054 [cs].
130. W. Zhang, W. Hua, and K. Stratos. “EntQA: Entity Linking as Question Answering”. 2021. arXiv: 2110.02369.
131. X. Zhang, J. Zhao, and Y. LeCun. “Character-Level Convolutional Networks for Text Classification”. 2015. arXiv: 1509.01626.
132. Y. Zhang, Z. Li, and M. Zhang. “Efficient Second-Order TreeCRF for Neural Dependency Parsing”. 2020. arXiv: 2005.00975.
133. Y. Zhang, Q. Xia, S. Zhou, Y. Jiang, Z. Li, G. Fu, and M. Zhang. “Semantic Role Labeling as Dependency Parsing: Exploring Latent Tree Structures Inside Arguments”. 2021. arXiv: 2110.06865.
134. X. Zhong, J. Tang, and A. J. Yepes. *PubLayNet: Largest Dataset Ever for Document Layout Analysis*. Aug. 15, 2019. https://doi.org/10.48550/1908.07836. arXiv: 1908.07836 [cs].
135. W. Zhou and M. Chen. “An Improved Baseline for Sentence-level Relation Extraction”. 2021. arXiv: 2102.01373.
136. W. Zhou, K. Huang, T. Ma, and J. Huang. “Document-Level Relation Extraction with Adaptive Thresholding and Localized Context Pooling”. 2020. arXiv: 2010.11304.
137. Z.-H. Zhou. “Multi-Instance Learning: A Survey”. In: *Dep. Comput. Sci. Technol. Nanjing Univ. Tech Rep* 1 (2004).