Daniel Gärber

Classification of Wikipedia Articles using BERT in Combination with Metadata

Bachelor's Thesis

to achieve the university degree of

Bachelor of Science

Bachelor's degree programme: Software Engineering and Management

submitted to

Graz University of Technology

Supervisor

Dipl.-Ing. Dr.techn. Roman Kern

Institute for Interactive Systems and Data Science
Head: Univ.-Prof. Dipl.-Inf. Dr. Stefanie Lindstaedt

Graz, August 2020

Affidavit

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly indicated all material which has been quoted either literally or by content from the sources used. The text document uploaded to TUGRAZonline is identical to the present master's thesis.

26.08.2020
Date

Signature

Abstract

Wikipedia is the biggest online encyclopedia and it is continually growing. As its complexity increases, the task of assigning the appropriate categories to articles becomes more difficult for authors. In this work we used machine learning to automatically classify Wikipedia articles from specific categories. The classification was done using a variation of text and metadata features, including the revision history of the articles. The backbone of our classification model was a BERT model that was modified to be combined with metadata. We conducted two binary classification experiments and in each experiment compared various feature combinations. In the first experiment we used articles from the category ”Emerging technologies” and ”Biotechnology”, where the best feature combination achieved an F1 score of 91.02%. For the second experiment the ”Biotechnology” articles are exchanged with random Wikipedia articles. Here the best feature combination achieved an F1 score of 97.81%. Our work demonstrates that language models in combination with metadata pose a promising option for document classification.

Contents

Contents

List of Figures

1 Introduction

Wikipedia is a great source of knowledge, but it must be ordered and classified to become useful to us. The complex structure of Wikipedia, which is steadily increasing, makes it difficult to assign articles to all of its corresponding categories. In this work we use machine learning to create a model that is able to classify Wikipedia articles from a specific category. Automatically assigning articles would help authors to find suitable categories for newly created articles and could overall improve Wikipedia's categorization system.

We focus on the classification of articles from the category "Emerging technologies". For our experiment we use past revisions of the articles where they have not yet been classified as "Emerging technologies". The control-set consists of articles from the category "Biotechnology" and a set of random Wikipedia articles. We perform binary classification of "Emerging technologies" and "Biotechnology" articles and then of "Emerging technologies" and the random articles. We use a combination of features that include the article content, the links to other articles, the categories the article is part of and the revision history. In our classification model the text features are processed with a pre-trained Bidirectional Encoder Representations from Transformers (BERT) [2] model, an influential deep language representation model. With BERT we can create contextualized sequence embeddings, which comprise the meaning of a sequence into a vector representation. These embeddings are combined with the metadata and classified in an additional output layer. The goal of this work is to find out if metadata can increase the model performance and if a BERT based approach for document classification is a viable option. As a baseline our model is compared to a traditional TF-IDF classifier.

Being able to detect which articles are part of the category "Emerging technologies" can also help detecting technologies that may play an important role in the future. This would help investors and businesses to predict future trends in the technology sector or stock market and could enable earlier business decisions. We hypothesize

that there exist patterns in the revision history of ”Emerging technologies” articles which would indicate a rise of interest in a technology.

2 Related Work

2.1 Background

This chapter first gives an overview of the prerequisites of machine learning in the context of natural language processing. Then the structure of Wikipedia and its articles is examined, in order to identify the important characteristics for classification.

2.1.1 Machine Learning (ML)

The aim of ML is to generate a computer program that is able to learn from experience and solves a specific task.

ML problems are usually divided into three main categories.

- Supervised learning
- Unsupervised learning
- Reinforcement learning

It should be noted that there exist other ML approaches that do not fit into these categories or are a combination of them. The basics of the three main approaches is now explained.

Supervised Learning

In supervised learning problems a model is given input data with the corresponding output data. The model learns from these examples, similar to humans, to map new unseen input to corresponding output. The input data in ML is typically a vector or matrix, and the individual values are called features. The output can be either a real

or categorical value, depending on the application. The process of learning from examples in ML is called training. The training data used in supervised learning is mostly labeled by humans, and typically the more labeled data is available the better the predictions of the model become. Popular supervised learning models include neural networks, support vector machines and logistic regression. Classic applications of supervised learning are object recognition or spam detection.

Unsupervised Learning

In unsupervised learning the model is given input data without the corresponding output data. This data is therefore unlabeled. The model now tries to find structures or patterns in the unlabeled data. Examples of unsupervised learning are clustering, dimensionality reduction and autoencoders. It plays an important part in systems that deal with large amounts of unlabeled data.

Reinforcement Learning

In reinforcement learning the model tries to improve in a task without supervision. It learns which actions are favorable and which are not. A popular example of reinforcement learning is Google's AlphaGo[1], which was the first program to beat a Go world champion.

2.1.2 Neural Networks

This section provides a short overview of neural networks and their influence on current trends in ML.

The structure and idea behind neural networks is loosely related to that of the human brain. The key components of neural networks are artificial neurons. These neurons receive one or more inputs and produce an output. Usually the inputs are weighted, summed together and passed through a non-linear activation function to produce the output. Neurons are stacked together to form layers. A neural network

[1] https://deepmind.com/research/case-studies/alphago-the-story-so-far (Accessed on: 2020-06-27)

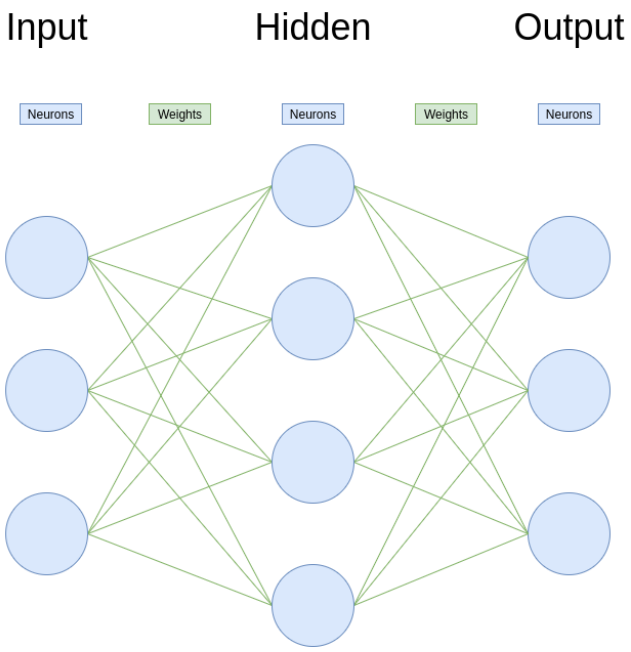

Figure 2.1: Basic structure of a neural network. The neural network is separated into an input, a hidden and an output layer. The layers consist of multiple neurons, each line represents a connection with an associated weight.

has an input layer, one or more hidden layers and an output layer. The output of neurons in a layer becomes the input of neurons in the following layer. The general architecture of a neural network is seen in Figure 2.1. Training is done by optimizing the weights of the neurons using an algorithm called backpropagation. If several hidden layers are used, we speak of deep learning (DL).

Neural networks have many advantages to traditional ML models. They are able to approximate complex functions, they generalize well and they are able to cope with large amounts of data. This comes, however, at some cost. Neural networks are computational intensive and it is difficult to understand their inner workings. In recent years neural networks had a rise in popularity in ML applications. The increase in computer resources made it possible to apply deep neural networks for many tasks. They have elevated the standard in various ML areas, including image, speech and text processing [13].

2.1.3 Natural Language Processing (NLP)

In this section NLP is introduced and an overview of its most important historical milestones is given.

NLP is the task of teaching a computer program to understand human language. There exist a wide area of NLP applications, including speech recognition, language generation, machine translation and text classification.

NLP problems were one of the first ML problems researchers tried to solve. In 1954 there was a famous attempt at automatically translating between English and Russia, which was called the Georgetown-IBM experiment[2]. First results of the project looked promising, but soon it became evident that the problem was far more complex than anticipated and further research was stopped. In the 1960s the computer program ELIZA[3] was developed, which was able to communicate with humans. It used the users input and special rules to construct responses, which created the illusion of a sentient conversation partner. ELIZA's understanding of human language however was very limited. Only in the 1980s ML began to play a larger role in NLP. Until then mostly handwritten and domain-specific rules were used in NLP-models. The reason for this was the increase in computation power and a better linguistic understanding. In recent years the focus in research has shifted to an unsupervised learning approach and neural networks, using vast text corpora that are available through the internet, for example Wikipedia.

As NLP has many application domains, this work focuses on the NLP task of text classification. Kowsari et al. [4] divide the process of text classification into four phases.

- Feature extraction
- Dimension reduction
- Classifier selection
- Evaluation

The input is typically a text data set. In feature selection, the data set is transformed into a structured feature space by cleaning the data and applying specific techniques like term frequency–inverse document frequency (TF-IDF) or converting the data

[2] https://en.wikipedia.org/wiki/Georgetown-IBM_experiment (Accessed on: 2020-06-22)

[3] https://en.wikipedia.org/wiki/ELIZA (Accessed on: 2020-06-22)

into a vector representation. Dimension reduction can then be applied to increase the performance of the system by only focusing on the significant features. Choosing the right text classifier depending on the application is difficult. Popular classifiers are logistic regression, naive Bayes classifier, support vector machines and neural networks. In the evaluation step the metrics of the model are analyzed, for example, accuracy or the F_1 score. In recent years neural network language models have set new standards in various NLP tasks. These will be explained in detail in the following pages.

2.1.4 Important Concepts in State-of-the-Art NLP Models

This section gives an overview of the foundational ideas behind state-of-the-art neural network NLP models according to Young [13]. First word embeddings and contextualized word embeddings are explained. Then the transformer architecture is introduced.

Word Embeddings

DL models make up the majority of current state-of-the-art NLP models. An important building block for this success is the use of word embeddings as features instead of hand-crafted features. Word embeddings are vectors that capture the characteristics of a certain word in a vector space that has far fewer dimensions than the original vocabulary. It follows the idea that words with a similar meaning are likely to occur in the same context. This is best explained with an example.

In Figure 2.2, we can see the visualized vector representation of the words “King”, “Man”, “Woman” and ”Queen”. The vectors “King” and “Man” have many similarities, as have the vectors “Man” and “Woman”. “Woman” and “King” have less similarities. In the vector space, words with a similar meaning are closer together. Therefore the vectors for “Man” and “King” are closer together than the vectors ”King” and ”Woman”. We can even do mathematically operations with these vectors. For example, the result of the operation “King” - “Man” + ”Woman” is the vector for the word “Queen”.

State-of-the-art word embedding models are DL models and are pre-trained on very large datasets. The training goal is usually to predict a word from the words

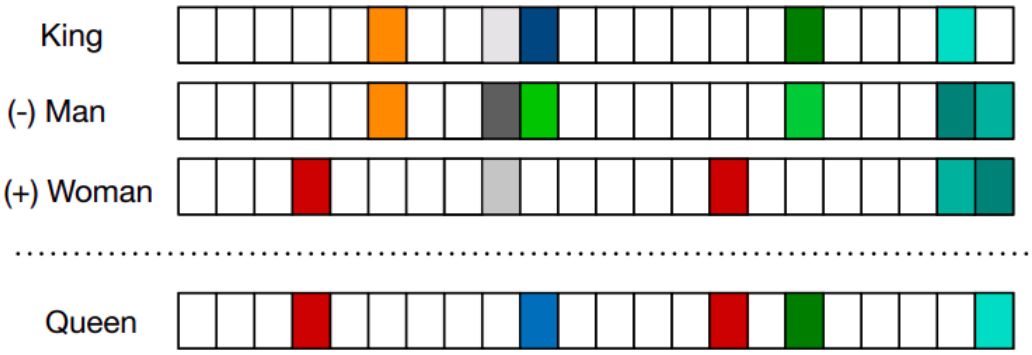

Figure 2.2: Visualization of word embeddings [13]. A word embedding is a vector representation of a word. Each value in the vector represents an abstract property of the meaning of the word. This enables mathematical operations on word embedding that can be interpreted in a semantic context.

surrounding it. A major limitation of word embeddings is that they are unable to represent phrases or a different meaning of a word depending on the context. For example, they are unable to differentiate between the meanings of the word "bank", as it can mean the bank of a river or the financial institution. A widely used word embedding model is word2vec[4].

Contextual Word Embeddings

Contextual word embedding models overcome these shortcomings, as the specific context of a word is also considered. One such model is Embedding from Language Model (ELMo). ELMo creates a different word embedding for each context the word is used in. So for the word "bank" of the example before there now exist two distinct vector representations. ELMo is a bidirectional model, which means it processes a sentence one time in forward and a second time in backwards direction. ELMo uses a recurrent neural network (RNN) architecture. In RNNs, the output of a token in a sequence is dependent on the previous calculations, which can be interpreted as giving each token context from the previous tokens in a sequence. This results in two hidden representations from each direction, which are then combined. By doing

[4] https://code.google.com/archive/p/word2vec/ (Accessed on: 2020-06-29)

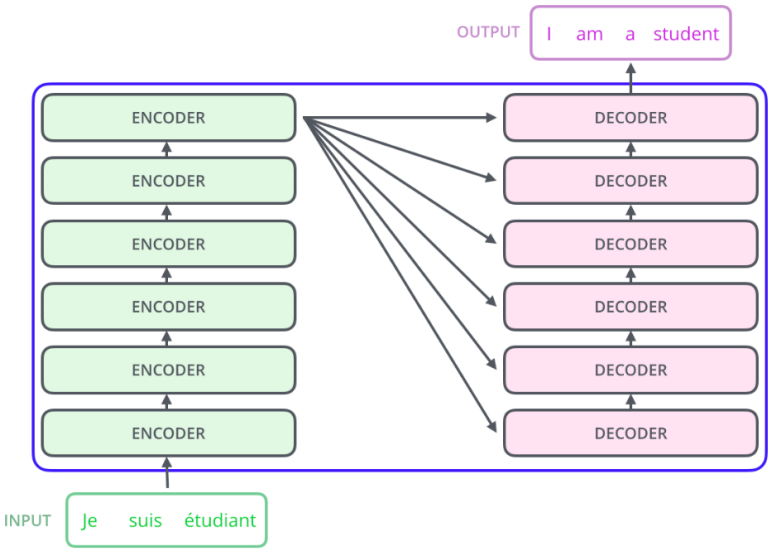

Figure 2.3: Overview of the transformer architecture. The encoder and decoder part of a transformer consist of 6 encoders and decoders. The intermediate representation made by the encoder is given to the decoder. Source: https://jalammar.github.io/illustrated-transformer/ (Accessed on: 2020-06-22)

this the model can take more of the surrounding context into consideration than traditional word embedding models. Bidirectional Encoder Representations from Transformers (BERT) is another model that takes context into consideration and outperforms ELMo in most tasks. BERT is based on the transformer architecture, which will be explained now.

Transformer

Transformers are the fundamental idea behind BERTs architecture. They were proposed in the influential paper ”Attention Is All You Need” by Vaswani et al. [11].

Transformers use an encoder-decoder design for sequence to sequence representation. The encoder creates an intermediate representation of a sequence of input words that is passed to the decoder, who in turn decodes the intermediate representation to a sequence of output words. A classic application of this process is text

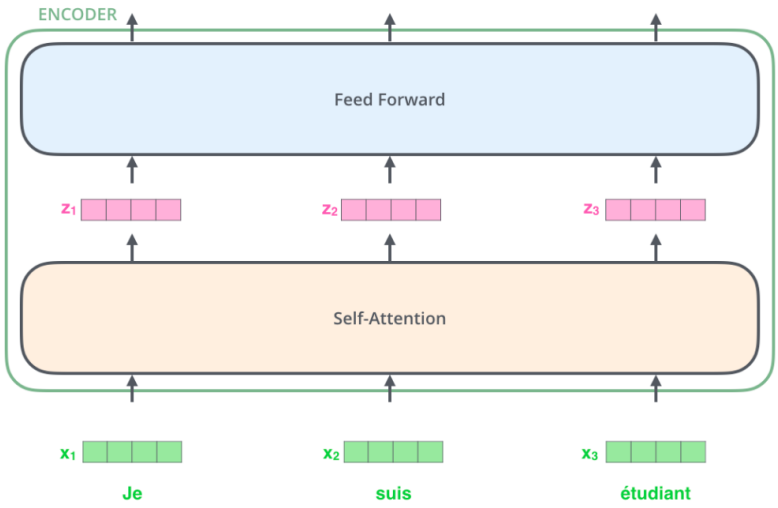

Figure 2.4: Single encoder in detail. A single encoder consists of two sublayers: The self-attention layer and a feed forward neural network. Each token gets individually processed. Source: https://jalammar.github.io/illustrated-transformer/ (Accessed on: 2020-06-22)

translation. For example, in French to English translation the text “Je suis étudiant” in French would first be encoded into an intermediate representation, which is then given to the decoder who outputs the English sentence “I am a student”. In the proposed architecture the encoder and decoder are made of 6 identical layers. The layers in the encoder and decoder are basically again encoders and decoders themselves, as seen in Figure 2.3.

As seen in Figure 2.4, each layer in the encoder consists of 2 sublayers: a self-attention layer and a neural network. Self-attention is a mechanism to focus on the important parts of a sequence. For example, in the sentence “The cat crossed a street because it saw a mouse”, the word “it” references “The cat”. In order to encode this sentence and its meaning correctly, the encoder must be aware of this connection. With self-attention, the encoder looks at all the other words when encoding “it” and calculates a score of how relevant each word is for the understanding of “it”. The second sub-layer is a fully connected feed-forward neural network. This network helps to make sense of the different attentions each word calculates to all other words in the sequence. The transformer architecture and attention greatly increase computational performance compared to an approach with RNNs, as the RNN computations are sequential, while the transformers operations can be

parallelized.

2.1.5 Wikipedia

This section gives information about Wikipedia, its article structure and features of articles.

The Wikipedia article about Wikipedia[5] states the following:

'*Wikipedia is a multilingual online encyclopedia created and maintained as an open collaboration project by a community of volunteer editors using a wiki-based editing system.*'

It is the largest online encyclopedia in the internet and was created in 2001 by Jimmy Wales and Larry Sanger. Wikipedia has a total of more than 53 million articles, from which 6.1 million articles are just of the English Wikipedia. There are currently more than 300.000 active volunteers contributing to the development of Wikipedia.

Structure of Wikipedia Articles

The structure of Wikipedia articles is explained in detail in Wikipedia's Manual of Style/Layout page[6]. An overview of the most important parts is given now.

Wikipedia articles can be divided into four elements.

- **Before the lead section**
  Metadata about the article and a short description, which is shown when searching for an article.

- **Body**
  Main part of the article. Consists of a lead section, a table of contents and the content.

[5] https://en.wikipedia.org/wiki/Wikipedia (Accessed on: 2020-06-24)
[6] https://en.wikipedia.org/wiki/Wikipedia:Manual_of_Style/Layout (Accessed on: 2020-06-24)

- **Appendices**
  References and further links.

- **End matter**
  Additional metadata, navigational links and a list of categories the article is in.

It is not mandatory for an article to include all elements of the proposed layout.

Most information about a topic is comprised in the body part. The body consists of a lead section, a table of contents and the content itself. The lead section[7] is a summarization of the most important points of a topic and serves as an introduction. Is has a special importance since most people looking for information on a Wikipedia article first read the lead section. After that comes the table contents[8], which is only present if the article has at least four headings. It helps navigating the content, which comes after. The content is the main source of information, and is typically divided into several sections with headings.

Wikipedia pages include links to other information sources on Wikipedia itself or to external sources. Wikipedia's internal links give the user a quick way to navigate to related Wikipedia pages. These links are called wikilinks[9]. A page also stores a list of all Wikipedia pages that link to this page[10].

Wikipedia uses a category system[11] to organize its pages. Pages are grouped together by categories and subcategories in a hierarchical tree organization. One page can be part of several categories. This enhances the navigability by helping users who just know the characteristics of a topic to quickly find matching articles. Wikipedia's recommendation is that every article should be part of at least one category.

[7] https://en.wikipedia.org/wiki/Wikipedia:Manual_of_Style/Lead_section (Accessed on: 2020-06-24)
[8] https://en.wikipedia.org/wiki/Help:Section#Table_of_contents_(TOC) (Accessed on: 2020-06-24)
[9] https://en.wikipedia.org/wiki/Help:Link#Wikilinks_(internal_links) (Accessed on: 2020-06-24)
[10] https://en.wikipedia.org/wiki/Help:What_links_here (Accessed on: 2020-06-24)
[11] https://en.wikipedia.org/wiki/Wikipedia:Categorization (Accessed on: 2020-06-24)

Features of Wikipedia Articles

For Wikipedia articles we can identify several types of features [3, 14, 6].

- **Content-based features**
  Words or n-grams in the article.

- **Link-based features**
  Links to and from other Wikipedia pages, including the categories the page is in.

- **History-based features**
  Article edit histories, for example the frequency of edits.

- **Usage-pattern features**
  Data about how often an article was viewed and what the previous viewed site or article was[12].

2.2 State of the Art

This section explains the state-of-the-art methods in text classification. First the architecture of BERT, an influential deep learning language model, is explained. Then it is described how BERT can be applied for document classification. Lastly the current state-of-the-art methods for classifying Wikipedia articles are discussed.

2.2.1 BERT

In this section the architecture and workflow of BERT is explained.

When BERT was first introduced by Devlin et al. [2] it achieved state-of-the-art results in 11 NLP tasks, for example, named entity recognition and question answering. Since then BERT has been outperformed in many tasks by other models, e.g. XLNet or RoBERTa [12, 5]. These models use a similar architecture to BERT

[12] https://meta.wikimedia.org/wiki/Research:Wikipedia_clickstream (Accessed on: 2020-06-24)

with certain improvements or optimizations. Therefore this section focuses only on BERT, since it set the foundation for many subsequent top performing models.

BERT is a deep language representation model. It is based on the transformer architecture, with the difference that it only uses the encoder stack, and instead of 6 encoders the standard BERT model has 12. The input to BERT is a sequence, which can consist of several sentences, with maximal 512 tokens. The first token of the input is always a special classification token called ”CLS”. If a sequence consists of more than one sentences, a ”SEP” token is used to separate them. Each word is first converted into a word embedding and combined with information about the position in the sequence. How the input is combined is visualized in Figure 2.5. The transformer architecture allows BERT to look at each word individually and check the context of the word in both directions. BERT outputs a representation for each token, including the ”CLS” token. The ”CLS” token can be seen as the aggregate sequence representation and can be used for classification tasks.

BERT is designed as a pre-trained model. In the paper they trained it on a large text dataset, consisting of the English Wikipedia and a vast book corpus. BERT uses two unsupervised pre-training tasks: masked language modeling (MLM) and next sentence prediction. In MLM, 15% of the input words are masked at random, and BERTs task is to predict these words. In next sentence prediction, BERT gets a pair of sentences from a document as input. In 50% of the cases the second sentence really comes directly after the first, and in the other 50% the second sentence is taken randomly from the document. BERT has to decide if the second sentence comes directly after the first sentence or not. A pre-trained BERT model can then be used for downstream tasks using fine-tuning. In the paper they proposed two models with different sizes, BERT_{BASE} and BERT_{LARGE}. They differ in the number of layers and size of hidden units, which results in about three times as many parameters in BERT_{LARGE}. BERT_{LARGE} outperforms BERT_{BASE} in all performed tasks, however the computational cost also increased. A variety of pre-trained BERT models have been made publically available[13].

[13] https://github.com/google-research/bert (Accessed on: 2020-06-28)

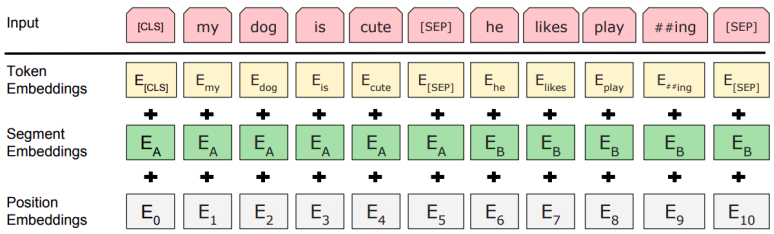

Figure 2.5: BERTs input representation [2]. Each token gets combined with its word embedding and additional information about its position in the sequence.

Document Classification with BERT

In this section state-of-the-art applications of BERT in the context of document classification are explained.

A pre-trained BERT model can be applied for downstream tasks which require language understanding, including text and document classification. Adhakari et al. [1] used BERT for document classification and achieved state-of-the-art results. This was achieved by connecting BERTs final hidden state of the special ”CLS” token to a fully-connected layer. BERTs architecture only allows a limited amount of tokens to be processed together, therefore they truncated documents that were too long. To cope with the computational overhead, they applied a technique called knowledge distillation. First they fine-tuned a large BERT model and then they distilled the knowledge of the BERT model into a much smaller and efficient long short-term memory (LSTM). This means the LSTM learns the representations of the BERT model, which were acquired prior. This smaller LSTM model overall achieved similar results to the basic BERT model while having far fewer parameters and a significantly smaller computational overhead.

Ostendorff et al. [7] used BERT for the classification of books. They had various kinds of data available: A short summary of the book, metadata and knowledge graph embeddings of the author. The knowledge graph embeddings were constructed with data from Wikipedia. The book title and the book summary were first processed with BERT to create a contextual representation. These representation were then concatenated with the metadata and author embeddings and served as input for a 2-layer neural network. Each unit in the output layer corresponds to a class label. The complete model architecture can be seen in Figure 2.6. Their results demonstrated

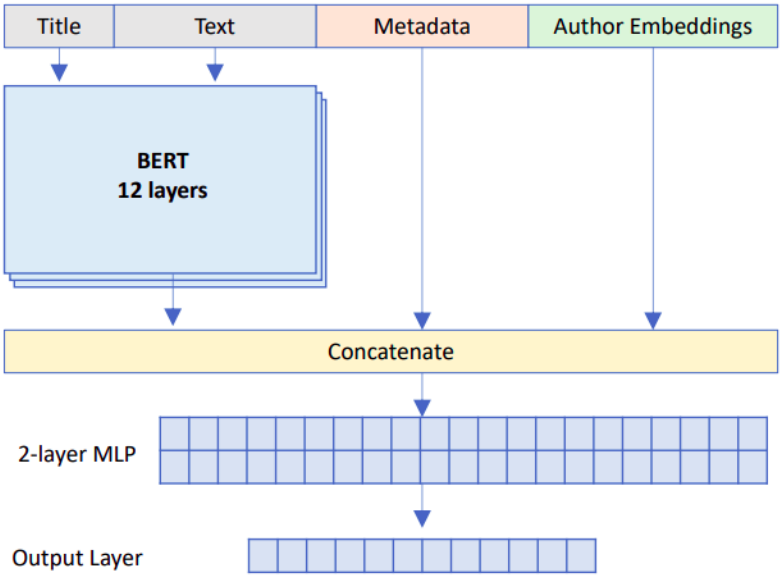

Figure 2.6: Architecture of the model used in [7]. Text data is first processed with BERT, then concatenated with metadata and author embeddings and fed into a neural network.

two points: Firstly, the additional metadata and author representation improved the overall result. Secondly, they compared the results achieved with BERT on document classification to a baseline done with logistic regression and TF-IDF, and BERT performed significantly better.

2.2.2 Wikipedia Article Classification

This section discusses different approaches of the classification of Wikipedia articles. It is a well-known problem, however it is not trivial. There have been various approaches, but more research is needed to achieve useable results.

Gantner and Schmidt-Thieme [3] showed that classifying Wikipedia articles is a feasible problem. They extracted content-based and link-based features from two different categories of the German Wikipedia corpus. The text was converted into a 1-gram representation and no stemming or stop-word removal was done. For classification they used support vector machines and achieved reasonable good

results. It should be noted here that this paper was published before neural networks models became state-of-the-art.

A difficulty with classifying Wikipedia articles is its detailed classification system and the data sparseness. Therefore many researchers use custom labels instead of the original Wikipedia categories. Suzuki et al. [10] classified articles to 200 fine-grained custom labels. Their dataset consisted of about 22.000 articles of the Japanese Wikipedia. The input features included the article title, headings, the article's first sentence and Wikipedia's categories the article was part of. An interesting feature they added is the way an article is linked from in other articles. For example, the article "Mount Everest" is linked from another article in the sentence "*... reached the summit of Everest for the twenty-first time ...*". The surrounding words provide additional information about the article. With word2vec they extracted this information by generating an embedding for each article using the surrounding text. The classification was done with a 2-layer neural network. To cope with the data sparseness, the labels were all jointly learned. Related labels formed clusters together and that improved the accuracy of the individual labels.

Shavarani and Sekine [8] extended this approach. They utilized multi-lingual features by using the content of the same article from different languages. The features were otherwise very similar to the ones extracted in [10]. They also labeled the data with a different label set than what Wikipedia is using. For classification, they extended the neural network model of [10] with an additional layer. They concluded that the best current models are neural networks, but they also struggled to achieve a good accuracy.

3 Method

3.1 Concepts

In this section the dataset is examined and the proposed model for our experiments is explained.

3.1.1 Dataset and Features

This section gives information about the type and structure of the used dataset.

Our dataset consists of Wikipedia articles of the categories ”Emerging technologies” (ET) and ”Biotechnology” (BT), and an additional set of random (R) articles. Both ET and BT are subcategories of the category ”Technology by type”.

It is important to note that usually not the most recent revision of an article is used. For articles of ET and BT we look at the point in the article’s revision history just before the article was classified as the respective category. The last revision before the classification is taken. For the R articles, a random revision is selected to mimic the work-in-progress state of the ET and BT articles.

For each article we have text and metadata features.

Text Features

- **Plain text**
  The unformatted article text without meta-information like links, pictures and supplementary sections (e.g. “Further reading”, “See also”).

- **Categories**
  Categories the article is part of.

- **Wikilinks**
  Outgoing links to other Wikipedia articles that are referenced in the article.

Metadata Features

- **Revision frequency per day**
  Calculated by dividing the number of earlier revisions by the number of days that passed since the creation of the article and the used revision.

- **Mean size increment per revision (MSI)**
  Mean increment of revision size between earlier revisions. The size is given in bytes.

- **Scaled revision peaks**
  Measures the amount of peaks in the revision history. We define revision peaks as timespans with a high number of revisions. The peaks are scaled by the total number of revisions.

- **Scaled editorial peaks**
  Similar to revision peaks, but focuses on the change of the article size per revision. We define editorial peaks as timespans where the article's size changed by a large amount. The peaks are scaled by the final size of the article.

- **Length of plain text**

- **Number of categories**

- **Number of wikilinks**

The revision frequency, the MSI, the revision and editorial peaks represent features of the revision history; the other features are extracted directly from the used revision.

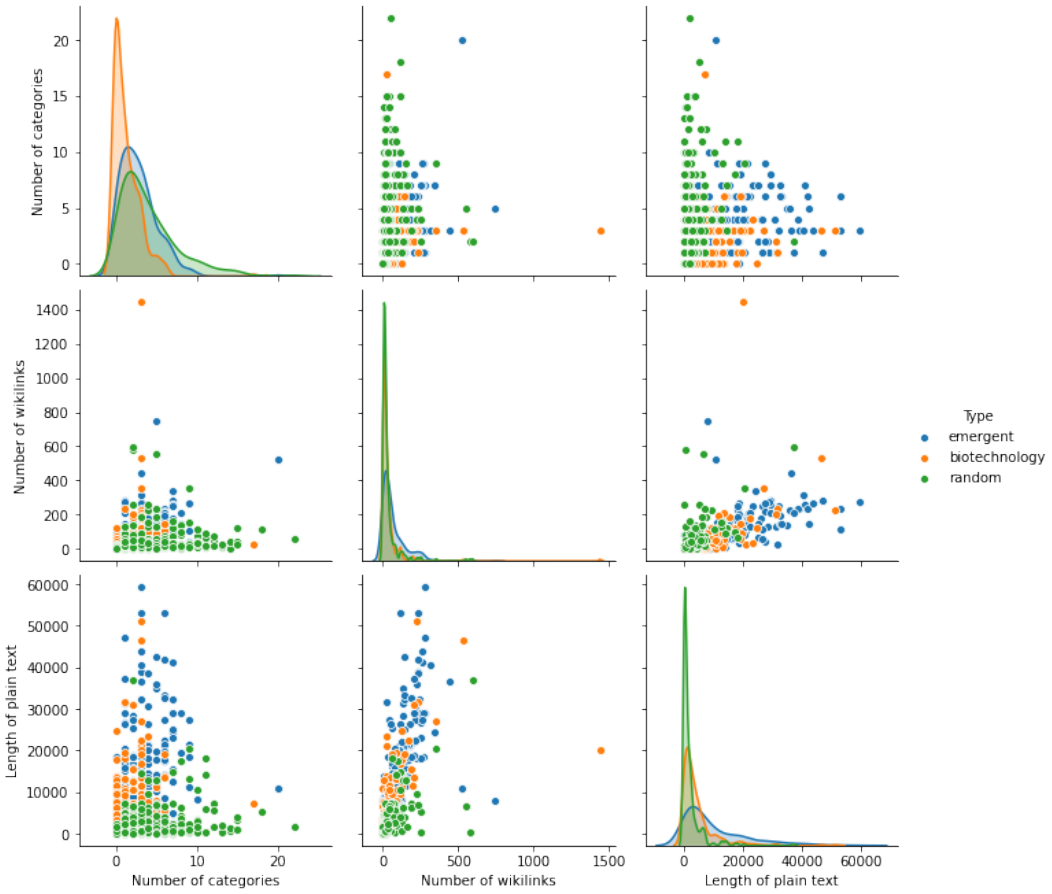

Figure 3.1: Visualization of number of categories/wikilinks and length of plain text for articles in comparison to their type. The diagonal graphs show the univariate distribution for that feature, all other graphs show a scatter plot of two combined features.

Figure 3.2: Visualization of the MSI, revision frequency, revision and editorial peaks in comparison to their type. The diagonal graphs show the univariate distribution for that feature, all other graphs show a scatter plot of two combined features. Some outliers had to be removed for the sake of clarity.

	Emerging technologies	Biotechnology	Random
Number of articles	319.00	292.00	300.00
Mean length of plain text	10220.10	4712.36	2159.35
Mean number of wikilinks	72.63	37.60	38.09
Mean number of categories	2.79	1.30	4.00
Mean revision frequency	1.63	1.96	0.58
Mean size increments	504.88	579.76	523.18
Mean revision peaks	0.64	0.52	0.52
Mean editorial peaks	0.58	0.31	0.31

Table 3.1: Statistics of the dataset.

Table 3.1 shows an overview of the features of the dataset. The articles from ET are generally longer, have more wikilinks and a higher frequency of peaks; however the size increments per revision and the revision frequencies are smaller than in BT.

In the Figures 3.1 and 3.2 the metadata features can be seen more clearly in context to their type. The diagonal graphs show a univariate distribution of that feature, all other graphs show a scatter plot of two combined features. Some feature combinations show a similar or random distribution, e.g. the MSI for BT and R have a nearly identical distribution, as can be seen in the top left graph of Figure 3.2. But there exist correlations, e.g. in Figure 3.1, the scatter plot of the length of the plain text and the number of categories approximate the type to some degree. It also has to be noted that in Figure 3.2 some outliers had to be removed for the sake of clarity. We choose to still include these outliers in our experiments, as our sample size is already small.

We had to make certain assumptions about the data to extract the metadata features. For example, the data about the peaks proofed difficult to extract because many articles only have a sparse or short revision history. Therefore the resulting values vary depending on the definition of what qualifies as a peak in the data.

3.1.2 Model Architecture

In this section the architecture of the classification model is explained.

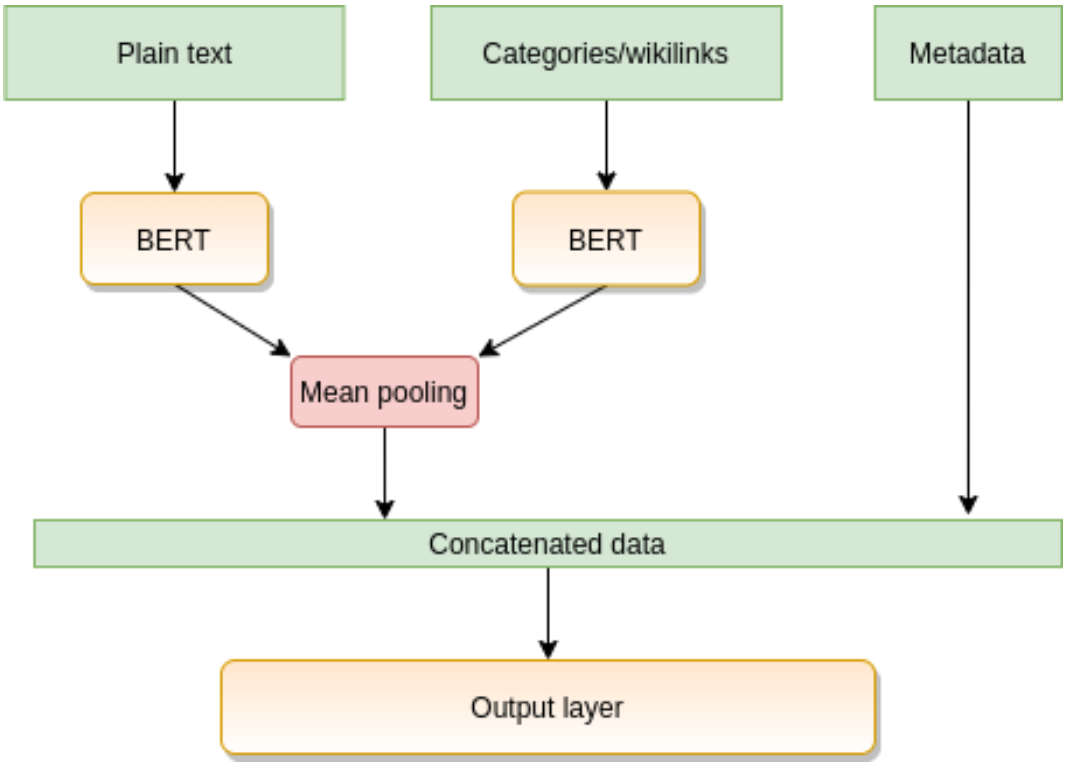

Figure 3.3: Model architecture. Textual features are first converted into sequence embeddings, averaged together and concatenated with the metadata features. The concatenated data is then classified using a linear output layer. The architecture is an adapted version of the architecture in [7].

The proposed architecture is similar to the model used in [7], the adapted architecture can be seen in Figure 3.3. The main idea is to combine text and non-text features for classification. We use the model for binary classification, first with the classes ET and R, and a second time with ET and BT.

The text features are processed with BERT. Due to BERTs limited input size, we use BERT twice: once for the plain text, and a second time for the concatenated categories/wikilinks (C/W). The plain text of many articles has to be truncated, but we hypothesize that most of the information about a Wikipedia article is found in the lead section at the beginning. As proposed by Sun et ali. [9], we compute the mean of both BERT outputs. This mean and the metadata are then concatenated and classified using a neural network. Our model is fine-tuned and tested on both classification tasks separately.

3.2 Implementation

3.2.1 Acquirement of Data

To our knowledge there does not exist a suitable dataset for our problem, therefore we had to collect it manually. This was done in Python[1] with the help of the MediaWiki API[2]. Unwanted pages in the categories were removed, e.g. subcategories, lists and user-pages. Since BT and ET are similar categories, they contain duplicate articles, which were also removed.

To extract the plain text, wikilinks and categories from the article data we used the Python package mwparserfromhell[3]. Articles that have always been part of either ET or BT, or had less than 20 plain text tokens, were not considered for our experiments.

[1] Version 3.6.9

[2] https://www.mediawiki.org/wiki/API:Main_page (Accessed on: 2020-07-29)

[3] Version 0.5.4 https://mwparserfromhell.readthedocs.io/en/latest/api/modules.html (Accessed on: 2020-07-29)

3.2.2 Preprocessing

This section explains which steps need to be taken to process text with BERT and the preprocessing of the metadata.

BERT

In our experiments we use the HuggingFace[4] BERT implementation. This includes the BERT model and the BERT Tokenizer.

Before the text can be processed with BERT, it has to be transformed into a format that BERT can work with. Several steps are needed to transform the text accordingly[5].

1. **Tokenization**
   The Tokenizer splits the words into tokens and converts these to a vector with pre-trained word embedding ids. For performance purposes we had to truncate our input to 200 tokens.
2. **Adding special tokens**
   The Tokenizer adds the [CLS]-token to the beginning and the [SEP]-token to the end of the word embedding id vector. These two tokens are needed for sentence classification.
3. **Padding and attention mask**
   Some vectors are shorter and have to be padded with [PAD]-tokens. Also an attention mask has to be created to prevent BERT from processing padded tokens.

The padded input vector and the attention mask are then ready to be processed with BERT.

[4] https://huggingface.co/ (Accessed on: 2020-08-1)
[5] https://jalammar.github.io/a-visual-guide-to-using-bert-for-the-first (Accessed on: 2020-08-1)

Metadata

The metadata included some outliers, especially for the revision frequency and the MSI. To still make this data usable, we scale it using the RobustScaler from scikit-learn[6]. This scaler is robust to outliers.

3.2.3 Classification Model

In this section the classification model is inspected in more detail.

Our model is implemented in PyTorch[7]. The internal structure is similar to the HuggingFace BertForSequenceClassification model[8], but instead of a single BERT layer our model consists of 2 BERT layers and incorporates additional data. The first BERT model processes the plain text, the second the C/W. The mean of both [CLS]-tokens is first fed through a dropout layer and then combined with the metadata. The [CLS]-token has a length of 768 and the metadata has a length of 7, which results in a combined vector of the length 775. This vector is then fed through a linear layer to compute the prediction.

The fine-tuning and validation code is based on the script "BERT Fine-Tuning Tutorial with PyTorch"[9] by Chris McCormick and Nick Ryan. Training is done with a batch size of 16 with 3 epochs. For optimization we use the HuggingFace Adam optimizer with a learning rate of 2^{-5} and a dropout probability of 0.1. These hyperparameters are proposed by Devlin et al. [2]. All experiments are run on a Tesla K80 GPU.

Listing 3.1 describes the most important parts of the proposed model and what operation are applied on the data during a forward pass in PyTorch pseudocode.

```
1    class ComBert():
2      def init(
3            self,
```

[6] Version 0.22.2

[7] Version 1.6.0

[8] https://huggingface.co/transformers/model_doc/bert.html#bertforsequenceclassification (Accessed on: 2020-08-6)

[9] https://colab.research.google.com/drive/1Y4o3jh3ZH70tl6mCd76vz_IxX23biCPP#scrollTo=6J-FYdx6nFE_ (Accessed on: 2020-08-6)

```
            dimensionsBert=768,
            dimensionsMeta=7,
            numLabels=2,
            dropoutProb=0.1):

        self.bertForPlainText = BertModel("bert-uncased")
        self.bertForCatwiki = BertModel("bert-uncased")
        self.dropout = Dropout(dropoutProb)
        self.classifier = Linear(
                            dimensionsBert+dimensionsMeta,
                            numLabels)

    def forward(
            self,
            plainTextIds,
            plainTextAttention,
            catwikiIds,
            catwikiAttention,
            metadata,
            labels):

        plainTextCLS = self.bertForPlainText(
                            plainTextIds,
                            plainTextAttention)

        catwikiCLS = self.bertForCatwiki(
                            catwikiIds,
                            catwikiAttention)

        meanCLS = (plainTextCLS + catwikiCLS) / 2
        meanCLS = self.dropout(meanCLS)
        CLSMetaCombined = cat(meanCLS, metadata)
        logits = self.classifier(CLSMetaCombined)
        return logits
```

Listing 3.1: Model architecture in PyTorch pseudocode. Describes the main parts of the model and shows the path the data takes on a forward pass.

4 Evaluation

In this section the evaluation methodology is explained and the results of the experiments are presented and discussed.

4.1 Methodology

Our experiments consist of two binary classification tasks. ”Emerging Technologies” (ET) articles are first classified against ”Biotechnology” (BT) articles and then against random (R) articles. The experiments where done using 10-fold stratified cross-validation. Stratified means that in each partition the percentage of each class stays approximately the same. We trained our model with different combinations of the features and compared the results. When using less features, e.g. only the plain text, the model parts for the other features are left out.

In the following results the mean of the cross-validation results is taken. The baseline is a simple logistic regression TF-IDF classifier from scikit-learn. The TF-IDF classifier only uses the plain text without additional metadata.

4.2 ET/BT Classification

In Table 4.1 the results for the ET/BT binary classification can be seen. The model with the plain text and the metadata had the highest F1, accuracy and precision score. The second best F1 score got the model with all three features. Models with the plain text yield better results than the ones without it. The plain text and metadata performed better than the plain text alone. The same is the case for categories/wikilinks (C/W) and metadata, which also performed better than just the C/W.

	Accuracy		Precision		Recall		F1	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std
Plain text	89.69	4.64	90.03	5.20	**90.58**	6.61	90.13	4.33
Plain text + metadata	**90.84**	4.63	**92.35**	3.90	89.97	7.18	**91.02**	4.70
Plain text + C/W + metadata	90.51	3.93	91.82	4.15	89.95	5.72	90.77	3.83
C/W	88.55	5.96	91.67	6.87	86.20	7.26	88.67	5.85
C/W + metadata	88.88	5.37	91.69	5.10	86.52	6.25	88.99	5.38
Baseline	88.38	7.14	88.83	6.65	89.02	8.24	88.84	6.87

Table 4.1: Mean of the cross-validation results for ET/BT classification in combination with different features.

4.3 ET/R Classification

	Accuracy		Precision		Recall		F1	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std
Plain text	97.25	1.73	96.16	3.59	**98.75**	1.61	97.39	1.60
Plain text + metadata	97.25	2.53	96.39	3.34	98.44	3.04	97.36	2.46
Plain text + C/W + metadata	**97.73**	1.92	**97.23**	2.28	98.42	2.26	**97.81**	1.88
C/W	96.28	1.88	95.46	2.49	97.50	3.23	96.42	1.85
C/W + metadata	95.95	3.00	95.96	3.33	96.22	3.29	96.06	2.94
Baseline	97.25	1.56	96.76	3.70	98.12	2.19	97.37	1.43

Table 4.2: Mean of the cross-validation results for ET/R classification in combination with different features.

In Table 4.2 the results for the ET/R Classification tasks can be seen. The best results on the F1, the accuracy and precision score were achieved using all 3 features. The TF-IDF baseline also performs very well on this task. Like in 4.2 the results obtained with the plain text were better than without it. Including the metadata alone does not improve the results, neither when used with the plain text nor in combination with C/W.

4.4 Discussion

For both tasks our model performed better than the baseline on the F1 score. The ET/R classifier overall performed better than the ET/BT classifier, which is probably because the articles in ET and R are more different than the articles in ET and BT. The plain text seems to be the most important feature in both tasks. The metadata and the C/W clearly influence the result, but it is not entirely conclusive in which direction. For the ET/BT task, the metadata in combination with the plain text performed best, whereas on the ET/R task the plain text alone performed better. The C/W data however improves the result on the ET/R task but worsens it on the ET/BT task. The positive or negative influence might be dependent on the specific task.

The standard deviation for the ET/BT task is quite high. We hypothesize that certain articles in ET and BT have a high significance for our model. In the ET/R task this is not the case, as the differences between ET and R articles are more distinct.

By combining all 3 features we beat the baseline in both tasks. On the ET/BT task our model is clearly better than the baseline, in the ET/R task there is only a slight difference. We showed with our result that a BERT based model can outperform the traditional TF-IDF approach on tasks involving long documents.

The small sample size makes it difficult to draw conclusions which combination of features is to be preferred, as no combination clearly outperforms the others. With a larger sample size the performance of the model could probably be improved and clearer patterns in the features would emerge.

The model architecture could also be refined for better performance. One weakness is the computation of the mean from the sequence embeddings of the plain text and the C/W. We hypothesize that by doing so useful information of each sequence embedding is lost. Also the classification layer has potential for improvement. We only used a single classification layer, and we hypothesize that with more layers the model might be able to capture the information in the text and metadata better. These optimizations will be left for future research.

To minimize target leakage we specifically used the revision of the article before it was added to the category. Since Wikipedia articles are continually improving, the category names ”Emerging technologies” or ”Biotechnology” can be found in some of the revisions before the article is classified. However we are confident that

Weight[?]	Feature
+5.051	quantum
+3.702	power
+3.584	vehicle
+3.291	meat
+3.246	light
+3.227	brain
+3.137	space
+3.029	air
+3.001	graphene
+2.980	talen
... 104393 more positive ...	
... 45588 more negative ...	
-2.725	pcr
-2.778	bio
-2.783	culture
-2.828	proteins
-2.935	sequence
-3.142	biological
-3.559	plants
-4.807	biotechnology
-5.277	protein
-5.534	dna

(a) ET vs BT

Weight[?]	Feature
+5.502	technology
+4.798	quantum
+4.319	used
+4.173	project
+3.967	energy
+3.908	cells
+3.746	brain
+3.562	display
+3.506	human
+3.372	using
... 118647 more positive ...	
... 31334 more negative ...	
-2.165	music
-2.172	song
-2.302	school
-2.312	north
-2.366	station
-2.477	village
-2.540	film
-2.591	south
-2.879	born
-2.923	county

(b) ET vs R

Figure 4.1: Most important terms with weights for both classification tasks from the TF-IDF baseline model. Terms in green are the most defining terms for ET, and the terms in red are the most defining terms for BT and R respectively.

our predictions were not significantly influenced by this. In Figure 4.1 the words with the highest and lowest weight of both classification tasks are shown. This data comes from the TF-IDF baseline model, but the significance of the terms to some degree also applies to our BERT based model. The term ”biotechnology” plays an important role in the ET/BT task, but this can be expected, as ”biotechnology” is a popular term in this topic. Other than that no traces of target leakage have been detected. In the ET/R classification task two terms with high weights for ET are ”used” and ”using”. Both words are probably used more frequently in a scientific context, but their importance could also be caused by the small sample size.

In the context of predicting emerging technologies for identifying promising stocks or emerging technology sectors, our model is not advanced enough to provide useful assistance. Unfortunately we were unable to find patterns in the metadata of the revision history of ET articles which would clearly indicate a rise in interest of a certain technology. However the overall results are promising enough to continue further research in this direction.

It must be noted that even though our model performs better than the TF-IDF baseline, the computational cost of training and applying BERT is far greater. Fortunately this is mitigated by the fact that most of the resources are only needed

during training. Still the computational cost could pose an obstacle in a practical application where speed is an important factor.

5 Conclusions

We classified Wikipedia articles of different categories with the help of text and metadata features. The different feature combinations show that metadata can increase the performance, however it depends on the task. In the binary classification of ”Emerging technologies” (ET) and ”Biotechnology” (BT) a combination of plain article text and metadata performed best. In the second experiment, the classification of ET and random articles (R), the combination of plain text, metadata and the wikilinks/categories (C/W) yielded the best results. In both tasks our BERT based model with included metadata performed better than the traditional TF-IDF approach. The metadata and C/W could at best only slightly improve the result of using just the plain text. Using the C/W and metadata alone yielded worse results in both experiments. Therefore we conclude that the plain text is the most important feature for our model. Despite our results the computational overhead involved in processing long texts with BERT is significant and more research needs to be done before a practical application can be considered.

5.1 Future Work

We identified several possible optimizations in our model which could be explored in future works. In the model architecture, more classification layers and a more sensitive approach when combining the sequence embeddings of different text features could prove beneficial. A major restricting factor of our experiments was the limited sample size. One possible solution would be to include the articles of the subcategories of ”Emerging technologies”, e.g. the category ”Artificial intelligence”. Another idea would be to include metadata features from other datasources besides Wikipedia, e.g. data from the Google search engine or various social media sites. Finding patterns there could not only improve the performance, but also help identifying emerging topics where none or only a rudimentary Wikipedia article

exists. Emerging technologies could possibly be detected earlier than with just the Wikipedia data. From a business perspective, the ability to detect emerging technologies as soon as possible is an important aspect, as it would give investors an edge over competitors.

6 Appendix

6.1 10-Cross-Validation Folds

This section shows all 10-cross-validation folds of the model with the best F1 score for the ET/BT and ET/R classification task respectively. In Table 6.1 are the results of the model for ET/BT classification using the plain text and metadata and in Table 6.2 are the results for ET/R classification using the plain text, metadata and C/W.

Fold	Accuracy	Precision	Recall	F1
1	87.10	96.15	78.12	86.21
2	83.61	86.67	81.25	83.87
3	88.52	87.88	90.62	89.23
4	91.80	90.91	93.75	92.31
5	100.00	100.00	100.00	100.00
6	88.52	90.32	87.50	88.89
7	93.44	91.18	96.88	93.94
8	95.08	93.94	96.88	95.38
9	88.52	93.10	84.38	88.52
10	91.80	93.33	90.32	91.80

Table 6.1: The results of all 10-cross-validation folds for ET/BT classification of the best model. The model used the plain text and metadata as features.

6 Appendix

Fold	Accuracy	Precision	Recall	F1
1	98.39	100.00	96.88	98.41
2	100.00	100.00	100.00	100.00
3	96.77	96.88	96.88	96.88
4	98.39	96.97	100.00	98.46
5	98.39	96.97	100.00	98.46
6	100.00	100.00	100.00	100.00
7	98.39	96.97	100.00	98.46
8	96.77	94.12	100.00	96.97
9	96.77	96.88	96.88	96.88
10	93.44	93.55	93.55	93.55

Table 6.2: The results of all 10-cross-validation folds for ET/R classification of the best model. The model used the plain text, metadata and C/W as features.

Bibliography

[1] A. Adhikari, A. Ram, R. Tang, and J. Lin. Docbert: Bert for document classification, 2019.

[2] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2018.

[3] Z. Gantner and L. Schmidt-Thieme. Automatic content-based categorization of wikipedia articles. pages 32–37, 08 2009.

[4] K. Kowsari, K. Jafari Meimandi, M. Heidarysafa, S. Mendu, L. Barnes, D. Brown, L. Id, and Barnes. Text classification algorithms: A survey. *Information (Switzerland)*, 10, 04 2019.

[5] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019.

[6] H. Moat, C. Curme, A. Avakian, D. Kenett, H. Stanley, and T. Preis. Quantifying wikipedia usage patterns before stock market moves. *Scientific reports*, 3:1801, 05 2013.

[7] M. Ostendorff, P. Bourgonje, M. Berger, J. Moreno-Schneider, G. Rehm, and B. Gipp. Enriching bert with knowledge graph embeddings for document classification, 2019.

[8] H. Shavarani and S. Sekine. Multi-class multilingual classification of wikipedia articles using extended named entity tag set, 09 2019.

[9] C. Sun, X. Qiu, Y. Xu, and X. Huang. How to fine-tune bert for text classification?, 2019.

Bibliography

[10] M. Suzuki, K. Matsuda, S. Sekine, N. Okazaki, and K. Inui. Neural joint learning for classifying wikipedia articles into fine-grained named entity types. In *PACLIC*, 2016.

[11] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need, 2017.

[12] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. Salakhutdinov, and Q. V. Le. Xlnet: Generalized autoregressive pretraining for language understanding, 2019.

[13] T. Young, D. Hazarika, S. Poria, and E. Cambria. Recent trends in deep learning based natural language processing, 2017.

[14] S. Zhang, Z. Hu, C. Zhang, and K. Yu. History-based article quality assessment on wikipedia. In *2018 IEEE International Conference on Big Data and Smart Computing (BigComp)*, pages 1–8, Jan 2018.