POLISH-JAPANESE ACADEMY
OF INFORMATION TECHNOLOGY

Machine Learning Tools and Techniques Supporting News Media Bias Analysis

PhD Thesis

mgr inż. Katarzyna Baraniak
supervisor: dr hab. Marcin Sydow

Polish-Japanese Academy of Information Technology
Warsaw, June 16, 2023

Contents

Part I

Background

Chapter 1

Introduction

The topic of media bias analysis gained popularity many years ago, and is still explored by scientist from many fields but still most of the work is done manually. In recent years a huge improvement of NLP techniques, especially introducing large language models, makes it possible to investigate the problem in automatic manner and analyse more aspects that are connected to bias. We wish to present several techniques that can help with analysis of the news articles textual content.

1.1 Aim and scope

The aim of this thesis is to propose methods that can help in news media bias analysis. In particular, we study entity-level sentiment analysis in short political news texts and persuasion techniques detection. In addition, we studied some helper techniques, that is: news articles similarity detection, entity timeline analysis and news source detection. We focus mainly on political news articles from English and Polish news outlets and we add experiments on Brazilian-Portuguese dataset.

1.2 Motivation

According to journalists' ethics, media outlets should present news that is fair and impartial. The readers should receive information that is clear, objective and complete to form their own opinions. In practice, readers are exposed to many types of bias. This problem is important especially for the digital media, since they have significant influence on public opinion. It is important to detect bias, especially in news portals, which should provide high-quality and fair content.

Media bias, especially political media bias, is usually presented in a simplified version as left, right, or center bias. In fact, it is much more complicated. Firstly, not all left- or right-biased media support the same view. Secondly, political bias is not the only category of bias in news.

For example the two popular portals, such `http://www.allsides.com` and `http://www.mediabiasfactcheck.com` gather information about news sites and

their bias annotated mainly by people.They present bias from different perspectives and types: political, ideological or bias in text. Political bias is usually considered as Left, Right, Center, when ideological refers to authoritarian vs. libertarian, traditional vs. progressive, elitist vs. populist etc. Bias in text can be classified as word choice, slant, flawed logic, sensationalism, opinion as fact and many more. We can observe that media bias is a complex topic that can be classified in different ways. It is important to prepare methods that help broader news analysis of different news types.

Automatic detection of media bias has useful applications in several fields. First of all, media bias analysis can improve the information transfer in democratic media and improving the quality of the public debate. However, often people read articles that present just one-side view which creates an incomplete picture of the world. This may happen due to few reasons for example personal preferences and subconscious choice of materials to read [1] which is called "confirmation bias", or algorithms in social media and search engines that prefer specific content [2] which is called "bubble filter". News searches and social media return information that are personalized based on the user, location, search history or other properties. Detection of news media bias may improve news searches so they produce more diverse results.

Moreover, media bias detection may improve the quality of democracy. If more people have access to more diverse information they are more resistant to manipulation and they are more open for different points of view. They can better assess and understand political and social situations. Media can be motivated to write about more diverse topics and society becomes better informed.

So far there are not too many automatic solutions for this problem. Many scientists from political or media science conduct their research based on manual analysis. Existing automatic solutions recognize the general bias or focus just on one type of bias. Most of them focus on the English language and situation in the USA where there are two main parties however media bias goes far beyond left or right bias. It may concern people, events, concepts etc. There are many countries in the world where politics is not strictly polarized and where there are more than two main parties.

There is a lack of automatic tools for Polish language and other underrepresented languages not only for media bias task. This may change soon, since language models like transformers make it easier and faster to develop new tools in NLP. However there is still a need to create new datasets, develop and test new models for many languages. This is also connected with previous point, in different countries and cultures media bias and political situation is not the same. Because of that it may be not enough to use the same NLP method but for different languages, it may appear that we need also different model of bias. Another motivation is that detecting media bias is an interesting research topic from a point of view of several disciplines: NLP, machine learning, social science and psychology.

To sum up, there is lack of good general automatic solutions and still a lot of interesting work has to be done.

1.3 Contents of the thesis

In the part I of this thesis we describe what is media bias and list its types. Next we explain theoretical background. We describe machine learning methods useful for news media bias analysis, starting from the basic ones to more advanced.

In the second part II multiple experimental results concerning media bias analysis are presented.

First, we present our preliminary experiments in section 4.1. Experiments presented in this section we previously described in the paper [3]. We proposed analysis of entities timelines as a method for recognising differences and anomalies in entities' occurrences. Using that method we can detect entities that take part in most popular events.

Next, we present experiments that we described in artcile [4]. We assume that for revealing bias in media it may be helpful to compare articles describing same events. For this reason we compared methods to find articles about the same topics in section 4.2. Also, we added experiments concerning detection of news sources in section 4.2.6.

Chapter 5 is dedicated to entity-level news analysis which is the main part of our experiments. First, in 5.2 we describe our bilingual "SEN"[1] dataset for entity-level news headlines analysis. It contains 1188 headlines in Polish and 2631 headlines in English from 8 and 5 news outlets respectively. In sections 5.3 and 5.4 we propose many experiments for entity-level sentiment classification on our dataset and two others: "PTB" (Brazilian-Portuguese language) and "news-mtsc" (English language). We created our own models and compare them with other state-of-the-art models. We created solutions that are comparable or better than state-of-the-art. Also, we discovered an issue that the sentiment that models learn is based on entities themselves more than the context (we call this phenomenon "entity sentiment bias") and propose a few techniques to overcome that problem 5.3.2. SEN dataset and some experiments were first presented in the article [5]. In this thesis we extended the experiments by the new analysis, models and results. In the last section 5.5 of this chapter we demonstrate an example application of entity-level sentiment detection for analysis of sentiment distribution in time through different news outlets.

Final chapter 6 of part II contains experiments concerning persuasion techniques detection in a multilingual dataset. We propose a multitask hierarchical BERT-based neural network to solve this problem. The solution was a part of semeval2023 competition and description of the network is accepted to be published in paper [6].

In part III we summarize the results of all experiments.

1.4 Main contributions

The main contributions of this thesis are:

- a novel dataset called SEN for entity-level sentiment analysis - it is the first dataset that covers two languages Polish and English and that was

[1] "Sentiment concerning Entity in News headline"

labeled both by researcher and crowdsourced annotators. The dataset is publicly available[2] and was downloaded 571 times so far [3]

- we present numerous experiments on SEN dataset and compare it with other available state-of-the-art models and datasets concerning:
  - extensive experiments concerning constructing deep learning classifiers detecting entity-level sentiment in news headlines, some of which beat some state-of-the-art[4] models of other researchers
  - analysis of sentiment entity bias of transformer models trained on the SEN dataset and several methods to solve this problem, consisting in replacing entity by mask token or by the type of the entity
  - enriching the vector representation of headline with Wikipedia context of the entity, that improves the performance of the classifier
- we created a neural network architecture for news articles persuasion techniques detection. It is a multitask hierarchical neural network based on large language models. Our experiments, performed on 7 different languages, show that our solution improves the BERT for token classification approach. Our model took part in prestigious international semeval 23 competition [5].

Additional contributions include:

- we propose a helper technique for finding articles on similar topics that can be potentially used for polarization and media bias analysis
- we indicate that it is easy to create a machine learning model that predicts the source of the article with high accuracy what may mean that the outlets are not objective. That means newspaper have some own specific style that can be further investigated regarding the presence of bias
- we present the possible analysis of the number of the particular entity occurrences in news articles

1.5 Software and Hardware

Experiments presented in chapters 4, 5and 6 were conducted using Python programming language. All the neural networks were created using PyTorch[6]. All transformer models were build with use of HuggingFace library[7]. Text preprocessing was done using mainly spacy[8]. Embedding models were implemented using gensim[9] or fasttext[10].

[2] https://zenodo.org/record/5211931
[3] 03.06.2023
[4] at the time when the experiments where conducted
[5] https://propaganda.math.unipd.it/semeval2023task3/index.html
[6] https://pytorch.org/
[7] https://huggingface.co/
[8] https://spacy.io/
[9] https://radimrehurek.com/gensim/
[10] https://fasttext.cc/

All the computations for neural networks took thousands of hours and were done on several servers and machines. Neural networks were trained on a server with 6 Nvidia Tesla K80 graphic cards[11] and on a Google Colab Pro [12] with limited access to V100, A100 or T4 Nvidia GPU. Some basic computations were done on local machines.

[11] In this place we would like to thank dr Radosław Nielek for sharing the server.
[12] `https://colab.research.google.com/`

Chapter 2

Related work

The media bias is a popular problem among researchers from many disciplines. In this chapter we briefly describe the most important works about media bias detection with machine leaning methods. We describe works that analyse general media bias, but also types of media bias and related tasks like tonality or sentiment analysis in the news, agenda and visibility detection or propaganda techniques classification.

2.1 General media bias

According to the work[7] media bias can be defined as the opposition of objectivity, neutrality, and fairness. The paper divides media bias into three types: visibility, tonality, and agenda. In this thesis we try to refer to each of these three types but first we wish to present works that concern general media bias detection.

There are different ways to define media bias and many proposed methods to analyse it. We present the most important ones.

In the work [8] the authors identify the news framing which is the way of presentation of news. They compare it to google trends and demonstrate how news framing influences the public attention. They used the concept of mean similarity of a corpus. The mean similarity is calculated on pairs of n articles by average cosine distance of DocVec representation of articles. They discovered that the public opinion change with the mean similarity.

In the paper [9] the authors describe some linguistic features that reveal the bias in a text. They refer to the form of verbs, part of speech tags and subjective words. Instead of news articles they used data from Wikipedia, however their results may be used also for other types of texts.

A related work concerning the usability of linguistic features in the task of detecting special form of bias related to the phenomenon of Web Spam is presented in [10].

An interesting approach is presented in paper [11], where the authors identify three roles of entities framing people in news articles. These roles are "hero", "villain" and "victim". The results are presented in a visual form to compare how entities are described in different articles.

The article [12] presents an approach of identifying bias through analysis of

mentions and quotations of politicians among different parties and in different periods of time.

Bias and its propagation is also investigated in social networks [13]. This work used Twitter data to identify bias in short texts and to analyse its propagation among the users.

In the work [14] the authors address detection of hyper-partisanship in news articles, i.e. classifying whether news is extreme left-wing or extreme right-wing using data annotated with distant-supervision. The best results, achieved by the ELMO based model, are about 82% accuracy and 80% f-measure.

[15] uses crowdsourcing to gather data and discusses automated methods to analyse the bias of various outlets concerning various issues but not the entities.

The work [16] provides extensive study on automatic media bias detection. Authors conclude that automatic tools are still not as good as human analysis.

2.2 Visibility and Agenda bias

In this section we wish to present works that correspond to the concept of visibility and agenda bias. Visibility bias is connected to the presence of entity in media. It refers to the number of entities mentions an it's exposure to the reader. Agenda refers to the scope of topics and aspects that are written about the entity. We decided to connect this two types of bias because the methods that we present later can support both of them.

First, we focus on entity extraction and then on finding similar articles by the topic.

Entity extraction and summarisation are problems which appear in many disciplines. The concept of event or entity timeline has been studied in some previous research, but we are not aware of any research publication that focused on Polish political news.

In the work [17] the authors present their method called 'Timemachine' which automatically generates timelines of events based on the knowledge base. Authors focus on three quality criteria: relevance of events to entities, temporal diversity, and content diversity.

In thearticle [18] authors describe their method of creating and analysing timelines of named entities. Their research is focused on entity evolution. As a part of their work, they analyse also political entities.

The work [19] presents a timemachine for Portuguese news. The results of their work are presented as an interactive web tool allowing searching and visualising news stories. Besides timelines, they present co-occurrences with an entity as an egocentric network.

An interesting approach is presented in a paper [20] where the authors describe a problem of connecting events described in news articles. They introduced a method using timelines to automatically find coherent chain linking events and topics described in articles.

Most of the current research on automatically generated timelines was focused on news and knowledge bases in English or other popular languages. In our research, we use only news in Polish, which have not been studied as carefully as other languages. Additionally, we are using the timeline to compare entities' occurrence between news portals.

There exist multiple approaches to identifying agenda bias. For most of them the first inevitable phase of bias identification is to find the pairs (or clusters) of similar articles, paragraphs or sentences. Then the way and scope of described event can be compared.

The approach of finding similar texts by using Siamese networks is described in [21]. Siamese networks describe how similar a pair of text documents are. These networks use the same architecture of network and feed two text documents as an input. Then, given such an input pair, an output in the form of the value representing the distance, for example Manhattan distance, between the two text documents from the output is calculated as a measure of (dis)similarity.

In the paper [22] the author describes document text representations and variety of similarity measures for text clustering. They include the measures like: cosine similarty, Euclidean distance, Jaccard coefficient, Pearson Correlation Coefficient and Averaged Kullback-Leibler Divergence. Then, based on the results of a clustering algorithm, there is made a comparison of the results on datasets including variety of news articles, academic papers and web pages.

Authors of the article [23] describe a similarity measure for news recommender systems. In this work there is made a comparison-based analysis of human judgement of similarity and some other measures such as: Lin and WASP measures.

Another work [24] presents research concerning articles on events. In particular it concerns tracking similar articles and clustering them to summarise the events under interest.

2.3 Tonality

The problem of entity-level tonality(or sentiment) in political news headlines is not as heavily studied as the related problem of target-based sentiment analysis concerning e.g. opinions on commercial products or services, however there exist some closely related works. Most of the works present some results on small datasets, datasets that are not publicly available or in other languages than covered in our research.

Although there are many approaches to the problem of media bias, to the best of our knowledge, there is no other similar work about detecting the entity-level sentiment of news headlines from readers perspective as we do. There are several works about detecting sentiment or polarity in the news presenting some different approaches. None of the work before is done on the Polish news articles.

The work [25] is the most similar to our experiments and presents a dataset, related to sentiment concerning entities in sentences from randomly selected news articles and consists of 3163 collected English-language sentences. The authors admit that a sentence is not enough to decide on the sentiment in the news and broader context is needed. In comparison to this dataset we created dataset of news' headlines, which are more standalone parts of articles that are always read by the users and that influence the sentiment of an entity more than random sentences of the article. Moreover, our SEN dataset is bi-lingual and we compare two annotator groups: researchers and crowdsourcing-based ones 5.2.

The continuation of the previous work [26] proposes an extended dataset for target depended sentiment classification in the news articles. Dataset is build

on English news articles and annotated by crowd-sorced workers. It contains about 11 thousand of targets that are annotated as positive negative or neutral. Annotations are made from the perspective of the text's authors. They proposed a neural network that uses sentiment dictionaries as external knowledge sources and additional mask to mark a target in a sentence.

A dataset concerning news sentiment is mentioned in work [27]. However, it is much smaller than our dataset and not available publicly. The work [28] presents datasets for brazilian portuguese news sentiment analysis focusing on brasilian elections.

Article [29] concerns the general impression of headlines with respect to a general topic rather than a target entity and report 74% and 79% of accuracy and F1, respectively. It uses only 100 headlines and 100 comments for evaluation.

Early work [30] addresses the problem of news articles' textual affect analysis. It uses an emotional lexicon to calculate the score of positive or negative words nearby the entity and thus seems to be not flexible and adaptive enough to political headlines. Another early work [31] provides only guidelines for sentiment annotation in the news articles but their dataset is not publicly available and no further analysis is done.

Simple, lexicon-based methods are used in [32] for measuring sentiment in citation presented in the news based on who is cited and how. The best results citations classification are around 82% of accuracy.

Work [33] studies information bias in the news as a problem of detecting biased phrases, their polarity and referenced entity. BERT-based model achieves 43% F1 at the sentence level bias and 18% at the token level.

A bit different problem but related to sentiment is 'stance detection' [34] in tweets. It's aim is to detect whether the author of a statement is supporting or is against a given concept. Entities are not a concern in the analysed tweets in this work.

In another work [35] authors present a task of flipping media bias in news headlines using encoders and decoders. If the headline was conservative, then the goal was to change it to more liberal and the opposite. Bias are then evaluated by humans.

The problem of bias may be approached in various ways, e.g. binary detection and neutralisation of biased phrases in Wikipedia articles and real media using BERT are presented in [36] achieving about 75% accuracy for detection.

Work [37] presents large scale sentiment analysis using expanded lexicons and score calculation.

As a novelty we introduce a Polish part of a dataset which is essential to extend the research on underrepresented languages. There are several datasets and works on sentiment analysis in Polish language: [38], [39], [40] but none of them concerns political news headlines.

Most of the works concerning sentiment analysis focus on reviews or tweets or financial news [41], [42], [43]. Financial news however are focusing on sentiment of companies, whether they are prospering or not in financial aspects.

We believe that the specificity of media outlets and in particular political news headlines deserves a separate approach with its very strong polarisation, dynamic character and specific language.

2.4 Persuasion detection

Detection of propaganda techniques is addressed in several works. The problem of persuasion detection is challenging due to the specificity of the language and quite large number of persuasion techniques what results in multiclass classification problem with unbalanced classes. There are a few approaches that try to solve the problem from different perspectives. Early work [44] uses transformers for sentence-level and fragment level propaganda detection.

The persuasion technique identification problem has been addressed as a problem of span and technique classification for the first time in work [45]. It was presented as a problem of multi-modal image and text input in article [46].

In contrast to the network presented in [47] our main aim is only persuasion technique detection, we do not use special tokens and we use one multitask network.

Authors of the work [48] propose creating ontology for propaganda detection in news articles.

The system to support analysis of propaganda techniques is proposed in [49]. It allows the user to explore articles, find spans of the propaganda techniques and summarize the statistics about these techniques.

The very recent work [50] studied detection of persuasion in Polish and Russian online news. They compare different granularity of labels and which level of text the classes are detected (words, sentences, paragraphs).

Persuasion is present not only in news but in our everyday conversations. A corpus for persuasion techniques detection in dialog is presented in [51].

A visual persuasion of social media content was study in [52]. They analysed persuasion of covid-19 related news shared on Twitter.

Chapter 3

Theoretical background

In this chapter, machine learning algorithms, that are suitable to the problem, are described. We describe both statistical and neural network models. The book [53] describes the most fundamental neural networks. There are also many works that summarize language models and methods used in natural language processing. For example [54] describes neural networks for machine translation which may be used also for other NLP tasks. The book [55] describes main concepts of natural language processing. [56] summarizes the most fundamental statistical learning concepts. All of these methods may be incorporated for media bias detection.

In the subsequent sections we overview the topics mentioned above.

3.1 Basic text representations

In this section we describe the basic concepts of text representation. We start from Bag-of-words, then tf-idf and finally we describe the concept of n-grams. More advanced text representations are described later in subsections 3.3.2, 3.3.9 and 3.3.10.

3.1.1 Bag-of-words model

Bag-of-words (BOW) model is a simple representation of a text as a set of words or tokens. For the first time BOW was mentioned in [57]. It is used mostly for feature generation and used as input for other models.

Before we describe BOW, we would like to define some basic concepts. A document is a text object. A corpus is a set of documents. Vocabulary is a set of tokens that appears in the whole corpus. Token can be a word, sub-word, character etc.

In BOW techniques a single document is represented as a binary vector[1]. The vector has length equal to the number of tokens in a vocabulary. Each coordinate of the vector represents one token from the vocabulary. If the document contains a particular token then there is "1" at the coordinate that corresponds to that term and "0" otherwise. The term order in the document is ignored.

[1] Or the number of occurrences of the term in the document in non-binary version of BOW

	document Bag of Words vector							
Document	this	dog	has	white	paws	cat	and	ears
This dog has white paws	1	1	1	1	1	0	0	0
This cat has white paws and white ears	1	0	1	1	1	1	1	1

Table 3.1: BOW model representation of a corpus with two documents. It is assumed that words are converted to lowercase and punctuation is omitted.

The BOW model can be illustrated on the example of a corpus containing 2 documents 3.1.1. Each column represents one term, each row represents one document. The disadvantage of this model is that it produces a sparse matrix representation of a corpus. We can observe that vectors represent the whole vocabulary of a corpus even if a term is not present in some documents.

3.1.2 Tf-idf

Tf-idf ("term frequency inverse document frequency") unlike BOW takes into account also the importance of a given term in a given document with respect to the whole corpus. The document vector, instead of "0" and "1", contains the weights of tokens on a scale from 0 to 1. The more often a term occurs in a document the higher value of tf-idf is. If token is popular in the whole corpus then value of tf-idf is lower.

The formula for tf-idf term is as follows:

$$tf\text{-}idf = tf(t,d) * idf(t). \tag{3.1}$$

$Tf(t,d)$ is a term t frequency in a document d with respect to number of other terms in this document, which can be expressed by:

$$tf(td) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}}, \tag{3.2}$$

where $f_{t,d}$ is a number of term t occurrences in document d and $\sum_{t' \in d} f_{t',d}$ is a sum of occurences of all terms in document d.

The second part $idf(t)$ measure how often the term t appear in a corpus:

$$idf(t) = \frac{|D|}{|d \in D : t \in d|}, \tag{3.3}$$

where $|D|$ is a number of documents in a corpus D. The denominator $|d \in D : t \in d|$ is a number of documents where term t appears at least once.

3.1.3 N-grams

N-gram is a sequence of n consecutive words or other items from a given text. One-word n-gram is called unigram, two-word is bigram, three words - trigram. Sentences or documents contain many n-grams. An example of n-grams from the document 'This dog is white.' is presented in table 3.1.3.

N-gram model is another possibility of input feature generation from text.The basic language models are based on the idea of n-grams. According to [58], the

n-gram type	n-gram tokens
unigram	This | dog | is | white
bigram	This dog | dog is | is white
trigram	This dog is | dog is white

Table 3.2: Example of n-grams of a sentence "This dog is white."

general aim of a probabilistic language model is to calculate the probability of a sequence of words.

$$P(W) = P(w_1, w_2, w_3, ..., w_k), \tag{3.4}$$

where W is a sequence of words and w_k is the k-th word in the sequence.

The special case of a probabilistic language model is N-gram model. It's aim is to calculate the probability of the k-th word based on a sequence of previous words.

$$P(w_k|w_1, w_2, w_3, ..., w_{k-1}), \tag{3.5}$$

The probability of a words' sequence of length K can be calculated using a chain rule:

$$\begin{aligned} P(W) &= P(w_1, w_2, \ldots, w_K) \\ &= P(w_1)P(w_2|w_1)P(w_3|w_{1:2})...P(w_k|w_{1:k-1})) \\ &= \prod_{k=1...K} p(w_k|w_{1:k-1}) \end{aligned} \tag{3.6}$$

In reality, there are too many possibilities of word occurrences in long sequences to calculate this probability. The idea of n-gram model is to approximate the probability of a next word given all previous words using a probability only of a few previous words instead. If N is the size of n-gram then the approximation of probability of a word w_k given it's all preceding words is:

$$p(w_k|w_{1:k-1}) \approx p(w_k|w_{k-N+1:k-1}) \tag{3.7}$$

To calculate the probability of a word using the unigram model, the following formula is used:

$$p(w_k|w_{1:k-1}) \approx p(w_k). \tag{3.8}$$

in case of the bigram model the formula is:

$$p(w_k|w_{1:k-1}) \approx p(w_k|w_{k-1}), \tag{3.9}$$

and for trigram model:

$$p(w_k|w_{1:k-1}) \approx p(w_k|w_{k-2:k-1}). \tag{3.10}$$

For example, when calculating the probability of the last word in a sentence "This dog is white." instead of computing conditional probability P(white|This dog is), bigram model calculates the approximation P(white|is) based only on the one preceding word. The trigram model calculates the approximation of third word based on two previous words, in this example P(white|dog is).

N-grams may be used not only in linguistics but also in other fields like medicine and genome sequencing.

3.2 Shallow machine learning

Next subsections cover basics of machine learning models that are used in later experiments logistic regression and support vector machines.

3.2.1 Logistic regression

Logistic regression [59] is binary classification algorithm that predicts the probability that an observation belongs to a particular class.

The probability is calculated by logistic function:

$$p(X) = \frac{e^{\beta_0+\beta_1 X_1+\dots\beta_p X_p}}{1+e^{\beta_0+\beta_1 X_1+\dots\beta_p X_p}}, \tag{3.11}$$

where $X = (X_1, X_2 \dots X_p)$ is input vector of length p and $\beta_0, \dots \beta_p$ is a learned coefficient. Coefficient can be estimated using maximum likelihood. The likelihood function is

$$l(\beta) = \Pi_{i:y_i=1} p_i \Pi_{i:y_i=0}(1-p_i), \tag{3.12}$$

where p_i is the estimated probability of i^{th} observation and y_i is the real answer. This function has to be maximized to find the best fit of the model.

3.2.2 Support vector machines

Support vector machines model is a generalization of the maximal margin classifier. It is a supervised learning model that may be used for classification, regression or outlier detection. SVMs were described in [60]. This classifier separates two classes with a hyperplane that has some margin. The hyperplane is chosen, so the separation distance is maximal.

In practice, it is rare to have perfectly separable data, so we use a soft margin classifier where some number of samples can lie inside the margin or on the wrong side of hyperplane. The kernel trick is applied to maximum-margin hyper-plane[61] for non-linear data.

As an example of the support vector machines with linear kernel for binary classification, see figure 3.1. It finds the solution to the following optimization problem

$$\underset{\beta_0,\beta_1,\dots,\beta_p,\epsilon_1,\dots,\epsilon_n,M}{\text{Maximize}} M \tag{3.13}$$

$$\text{subject to} \sum_{j=1}^{p} \beta_j^2 = 1 \tag{3.14}$$

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \dots + \beta_p x_{ip}) <= M(1-\epsilon_i) \tag{3.15}$$

$$\epsilon_i \geq 0, \sum_{i=1}^{n} \epsilon_i \leq C, \tag{3.16}$$

where M is the width of the margin that is maximized by svm, C is a non-negative tuning parameter,$\beta_0, \beta_1, \dots, \beta_p$ are parameters of hyperplane, $\epsilon_1, \dots, \epsilon_n$ are slack variables that specify if the observation i is on the right side of the margin. The last parameters allow some training samples to be on the wrong side of the hyperplane or margin to prevent overfitting and create greater margin.

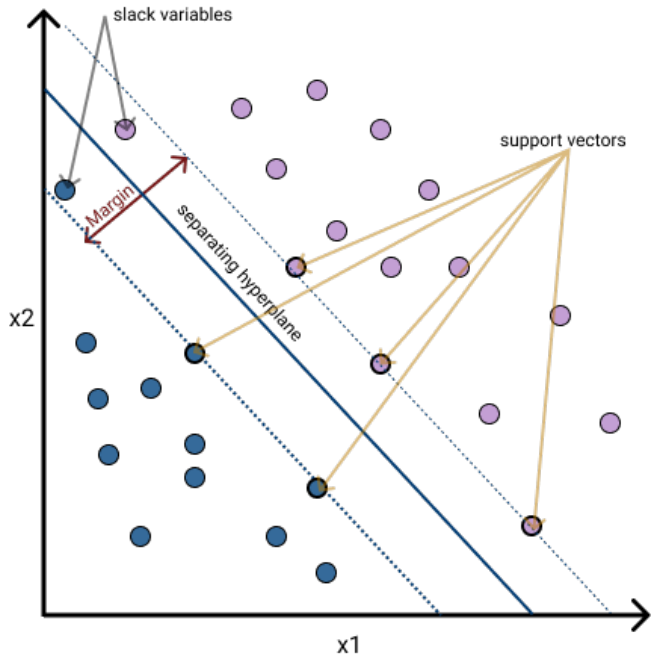

Figure 3.1: Support vector machines with a linear kernel separating 2 classes. SVMs maximize the margin between classes marked with blue and pink. The margin lies in variables that are called support vectors. Exactly in the middle distance from both margins there is a separating hyperplane. Sometimes slack variables are allowed, so the margin or even hyperplane do not separate them correctly.

After solving the optimization problem, the observation x is classified based on the sign of the equation:

$$f(x) = \beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p \tag{3.17}$$

In general case, when the kernel is non linear, the decision function has the form:

$$f(x) = \sum_{i=1}^{N} \alpha_i K(x, x_i) + \beta_0 \tag{3.18}$$

where n is a number of support vectors, α and β are parameters to be learned. K is a kernel function.

A kernel function maps input vectors into another vector space so that classes are easily separable. It is not necessary to calculate dot product in another vector space thanks to a kernel trick. Support vector classifier may have various forms of kernels. The most popular kernel functions are:

linear kernel

$$K(x_i, x_{i'}) = \sum_{j=1}^{p} x_{ij} x_{i'j}, \tag{3.19}$$

polynomial kernel

$$K(x_i, x_{i'}) = (1 + \sum_{j=1}^{p} x_{ij} x_{i'j})^d \tag{3.20}$$

radial kernel

$$K(x_i, x_{i'}) = exp(-\gamma \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2). \tag{3.21}$$

3.3 Neural networks

In this section we describe neural networks that can be used in tasks related to natural language processing. We start from perceptron, then we move to feed-forward neural networks, word embeddings, basic recurrent neural networks followed by LSTMs and GRU networks. Then we also mention a convolutional neural network. Next, we describe the autoencoders and the attention, that are used to create transformers, described in the next subsection. Finally, we describe BERT, one of the large language models.

3.3.1 Feed-forward Neural Networks

The basic form of a neural network is a single perceptron. The perceptron consists of the input vector x, the weight vector w, the activation function f and the output y. The signal of the perceptron is $net = w^T x + b$. The output of the perceptron is calculated as :

$$y = f(net) = f(w^T x + b) \tag{3.22}$$

The activation function of the perceptron is a step function. In that case the output is:

$$y = \begin{cases} 1 & net > 0 \\ 0 & \text{otherwise.} \end{cases} \tag{3.23}$$

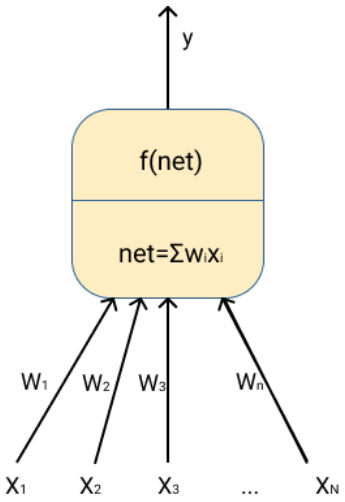

Figure 3.2: A simple neuron model with the input vector $x_1, ..., x_n$, the weight vector $w_1, ..., w_n$, the activation function f and the output y. Net is a dot product of the weight vector and the input vector.

Single perceptron is a binary classifier, that can distinguish only linearly separable classes. From this point the idea of neural networks, that are able to solve much more complicated tasks is developed.

There exist several activation functions that can be used instead of step function. The model with continuous and non-linear activation function is called a neuron or sometimes a unit. Some other types of activation functions are:

- sigmoid $f(x) = \frac{1}{1+e^{-x}}$
- hyperbolic tangent $f(x) = \frac{2}{1+e^{-x}} - 1$
- linear $f(x) = x$
- Rectified Linear Unit (ReLU) $f(x) = max(0, x)$

Model of the single neuron is presented in figure 3.2.

Multi-layer perceptron (MLP) or feed-forward networks (FFN) [53] are neural networks that can have many layers. A layer of a neural network consists of one or more neurons connected to previous layer or input. The last layer is called output layer and any other layer is hidden layer.

For instance if MLP has one hidden layer and one output layer the equation for the output calculations are:

$$h = f_1(W_{xh}x + b_h) \tag{3.24}$$

$$y = f_2(W_{hy}h + b_y) \tag{3.25}$$

where x is input vector, h is hidden layer, y is output, W_{xh}, W_{hy} are corresponding weight matrices, b_h, b_y are biases and f_1,f_2 are activation functions.

Hidden layers are followed by non-linear activation function, usually ReLU. An activation function of the output layer depends on the task being solved.

For example, a sigmoid function can be used for a binary classification or a multilabel classification. For multiclass classification the softmax function is usually used. Then the equation for the i_{th} neuron of the last layer is:

$$y_i = f(net_i) = \frac{e^{net_i}}{\sum_{j=1}^{J} e^{net_j}} \tag{3.26}$$

where J is the number of classes, $net_i = w_{iy} + b_{iy}$, w_{iy} and b_{iy} are weight vector and bias of i_{th} neuron, when $\sum_{j=1}^{J} e^{net_j}$ is a sum for all neurons in the output layer. For a regression task a linear activation function can be used.

In general, feed forward neural networks can be thought of as a chain of functions that takes some input and calculates some desired output. Composition of simpler functions can represent some more complicated functions. Neural networks can approximate any kind of function and are considered as a universal function approximation.

The Universal Approximation theorem [62] states that for continuous, bounded and monotonic sigmoidal function $\sigma(i)$ such that:

$$\lim \sigma(t) = \begin{cases} 1 & t \to +\infty \\ 0 & t \to -\infty \end{cases} \tag{3.27}$$

any continuous, multivariable, real function $f(x_1, x_2, ...x_p)$, defined within $[0, 1]^P$, whenre $x_1, x_2, ...x_p$ are input variables, and for any $\epsilon > 0$, there exists such integer N and set of real constants a_i, b_i, w_{ij} so that if

$$g(x_1, x_2, ...x_p) = \sum_{i=1}^{N} a_i \sigma(\sum_{j=1}^{P} w_{ij} x_j + b_i), \tag{3.28}$$

then the following inequality is satisfied

$$|f(x_1, x_2, ...x_p) - g(x_1, x_2, ...x_p)| < \epsilon \tag{3.29}$$

That means any function can be approximated by one layer neural network with finite number of neurons. However, adding more layers to neural network can result in lower number of neurons in total .

The loss function is used to evaluate solutions during training. The loss function calculates the cost of the solution, which can be interpreted as an amount of error concerning the answer of network and the correct answer. L1 and L2 loss functions are the most popular. L1 is the absolute difference between the estimated value p and the real target value y:

$$Loss(y, p) = |y - p| \tag{3.30}$$

L2 is the squared difference between the true y and the estimated target value p:

$$Loss(y, p) = (y - p)^2 \tag{3.31}$$

These two functions are usually used for a regression problem. Popular loss function for a classification problem is cross-entropy. When the number of classes is equal to 2 then the formula is as follows:

$$Loss(y, p) = -(ylog(p) + (1 - y)log(1 - p)) \tag{3.32}$$

and if there are more classes than 2:

$$Loss(y, p) = -\sum_{c=1}^{M} y_c log(p_c), \tag{3.33}$$

where M is the number of classes, p is the probability of a class, y is the target class indicator that can have two values: 1 if observation belongs to class c and 0 otherwise.

Gradient-based optimization algorithms are used to train a neural network. These algorithms minimize the loss function of the network. An example of gradient method is gradient descent. In this algorithm the weights are updated by a small value in the opposite direction to the gradient of the loss function. The formula for the change of the weight w_{ij} with a gradient descent method is:

$$\Delta w_{ij} = -\eta \frac{\partial E}{\partial w_{i,j}}, \tag{3.34}$$

where w_{ij} is the weight vector of i^{th} neuron in the j^{th} layer, $E = Loss(y, p)$ is the error of the network and η is learning rate that scales the gradient and controls the amount of change.

Then:

$$\frac{\partial E}{\partial w_{i,j}} = \frac{\partial E}{\partial net_i} \frac{\partial net_i}{\partial w_{i,j}}, \tag{3.35}$$

is error signal of the i^{th} neuron in the j^{th} layer .

During training the weights and biases of all neurons in all layers are fitted using backpropagation method. First, the signal errors of the output layer neurons are calculated and their weights are updated. The hidden layers are updated based on the error signal from the output layer or the layer that is next to them. The error is propagated from the output through all hidden layers to the first layer.

3.3.2 Word Embeddings

Words cannot be inserted into the network directly. We need some digital representation of them that can be understandable by neural networks. At the early stages, words were represented as one-hot vectors, bag of words or tf-idf representations which was easy and unambiguous representation but suffers from many drawbacks. First, in large corpus, vectors for documents have extremely large dimensions and use a lot of memory. Sparse matrix that is created to represent documents in corpus, may lead to increased space and time complexity. This kind of problem is similar to the problem of curse of dimensionality. Moreover, there is a possibility that the set of words in a test sample is totally different from samples that model seen in a training process, which is called semantic gap.

Another problem of this representation is that it does not cover context of the words or their semantic similarity. The words 'cat' and 'kitty' will be treated as completely different words despite they are strongly related. Another problem is homonyms, for example the word "book", in sentences "I read a book" and "I book a flight", should have different vectors as the meaning is different.

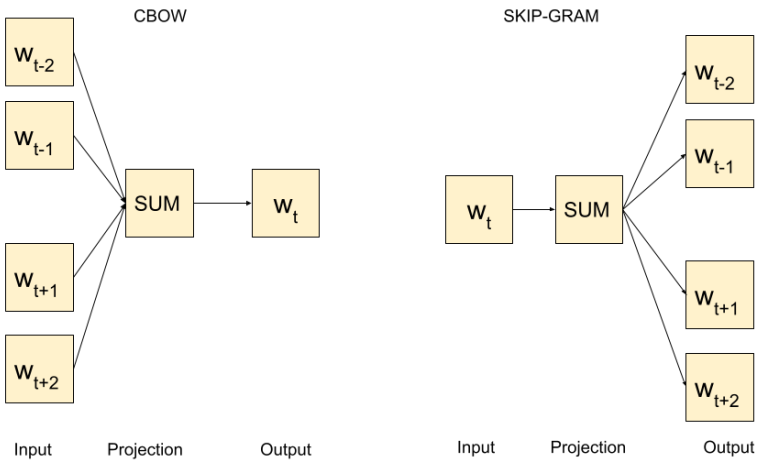

Figure 3.3: Architecture of word2vec cbow and skip-gram models. w_t is a target word, $\{w_{t-2}, w_{t-1}, w_{t+1}, w_{t+2}\}$ are context words.

The solution for the first problem with semantic similarity are word embeddings. In general such word embedding is a vector that represents semantic meaning of a word as a real vectors. There are many methods of creating word embeddings. For the first time they were introduced in [63] as neural language models.

The next version of word embeddings are word2vec [64] [65]. There are two architectures of word2vec : CBOW and Skip-gram. Both models are two-layer neural networks. The input is a large corpus and the output is a vector space that assigns vectors to each unique word in the corpora. In CBOW approach the neural network takes as input some window of words and tries to predict a target word that fits to the context of given words. The window slides through the whole corpus and learns vectors for each word. The order of context words does not matter. The Skip-gram model tries to predict the surrounding window of context words from a single word as an input. This model assigns weights to context words. The more distant the word is the lower weight it has. In both method the embedding is created from a hidden layer weights. Both architectures are presented in figure 3.3.

The resulting word embeddings contains such vectors that the words which are more semantically and syntactically similar, have vectors that are closer to each other and more dissimilar words have farther vectors.

A little bit later glove [66] vectors were introduced. It is a regression model that combines global matrix factorization and local context window methods.

Next type is an embedding of sub-words known as fastText [67]. These word vectors use part of words to create embeddings of the whole words, so they reduce a problem of unknown words and word's inflection.

Word embeddings are not enough to represent the complexity of language. They can only create one embedding per word or sub-word even if this word may appear in many different contexts with different semantic meaning. They have also some problems with unknown words, partially solved by fastext.

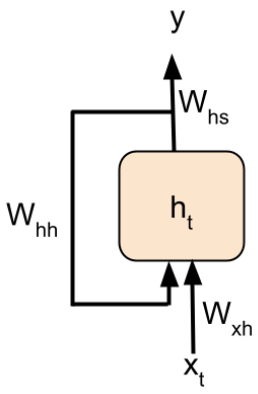

Figure 3.4: RNN cell. W_{xh} is a weight matrix of input, W_{hh} is a weight matrix of previous hidden state W_{hs} is a weight matrix for a current hidden state.

3.3.3 Recurrent Neural Networks

Another type of neural networks are recurrent neural networks (RNN) [68]. This type networks are able to precess the sequence of any length.That makes them more capable for language processing than feed-forward networks.

The input of such network is a sequence $X = \{x_1, x_2, x_3, \ldots, x_t\}$ where each element x_t is an input at timestep t. The network calculates the hidden state h_t and the output y. The idea is to add connection, which passes the value of the previous hidden state h_{t-1} to the current neuron's input. Equations to calculate basic RNN's output at timestep t are:

$$\begin{aligned} h_t &= g_1(W_{xh}x_t + W_{hh}h_{t-1} + b_h) \\ y_t &= g_2(W_{hs}h_t + b_s) \end{aligned} \tag{3.36}$$

where h_t is the hidden state at time step t, y is the output of network, g_1 and g_2 are activation functions, h_{t-1} is the previous hidden state, W_xh is weight matrix for input, W_{hh} is weight matrix for previous hidden state, W_{hs} is weight matrix for current hidden state, b_h and b_s are corresponding biases.

The illustration of the RNN's neuron is presented in picture 3.4. RNN can be also presented unfolded as a repetitive structure similar to chain of events.

RNNs may be categorised on input and output type, suited to the task it performs:

- vector-to-sequence - in this type an input is a single element, then the output is generated at each timestep and used as input for the next time step. This kind of input and output representation can be used for sequence generation task, for example in text generation task

- sequence-to-vector - the input is a sequence and there is only one output. This architecture is used for sequence classification tasks like sentiment analysis

- sequence-to-sequence with input and output of the same length - the network produces as many outputs as the input length. This is mostly used for sequence token classification tasks like named entity recognition
- sequence-to-sequence with input and output of different lengths - the input shape and output shape doesn't have to be the same for the task where we generate a new sequence based on the input sequence, for example machine translation

Recurrent neural networks are trained using a backpropagation through time [69]. In the basic version of this method the network is unfolded, so it contains a sequence of t inputs and t outputs. Every instance of unfolded network shares the same parameters. The error is calculated at each time step and backpropagated through all previous instances of unfolded network. Calculated weights changes are summed together. Then the weights are updated.

3.3.4 LSTM

The problem of basic RNNs is a vanishing gradient and an exploding gradient. The vanishing gradient is when we run a backpropagation method to train neural network and gradient gets smaller and smaller, and the exploding gradient is the opposite. The long short-term memory (LSTMs) neural networks [70] are a special kind of networks designed to reduce the problem of vanishing gradient. In addition to hidden state they also contain memory cell and forget cell. This two cells are not susceptible to vanishing or exploding gradient. As a consequence they are able to catch long term dependencies better than simple RNNs. The functions for the LSTM cells are as follows:

$$\begin{aligned} i_t &= \sigma(W_{ii}x_t + b_{ii} + W_{hi}h_{t-1} + b_{hi}) \\ f_t &= \sigma(W_{if}x_t + b_{if} + W_{hf}h_{t-1} + b_{hf}) \\ g_t &= \tanh(W_{ig}x_t + b_{ig} + W_{hi}h_{t-1} + b_{hg}) \\ o_t &= \sigma(W_{io}x_t + b_{io} + W_{ho}h_{t-1} + b_{ho}) \\ c_t &= f_t \odot c_{t-1} + i_t \odot g_t \\ h_t &= o_t \odot \tanh(c_t) \end{aligned} \tag{3.37}$$

where h_t is the hidden state at the time step t, c_t is the cell state that preserves LSTM from the exploding gradient, f_t is the forget gate which sets the weight of the previous cell state from 0 to 1 using a sigmoid function, i_t is the input gate which controls how much of the new input is let in using sigmoid function, g_t is the cell gate which is responsible for an update of the information coming from the input, o_t is an output gate that controls the output of the hidden state. The variable x_t is the input at the time t. For each gate W_i, W_h and b_h denote input weights, hidden state weights and bias respectively.

The model of the LSTM cell is presented in 3.5. There are various types of LSTMs, usually with modifications of the forget gate or the output gate.

3.3.5 GRU

Another variant of RNN is Gated Recurrent Unit Network (GRU) [71]. This architecture is easier to train than the LSTM network, without loses on the

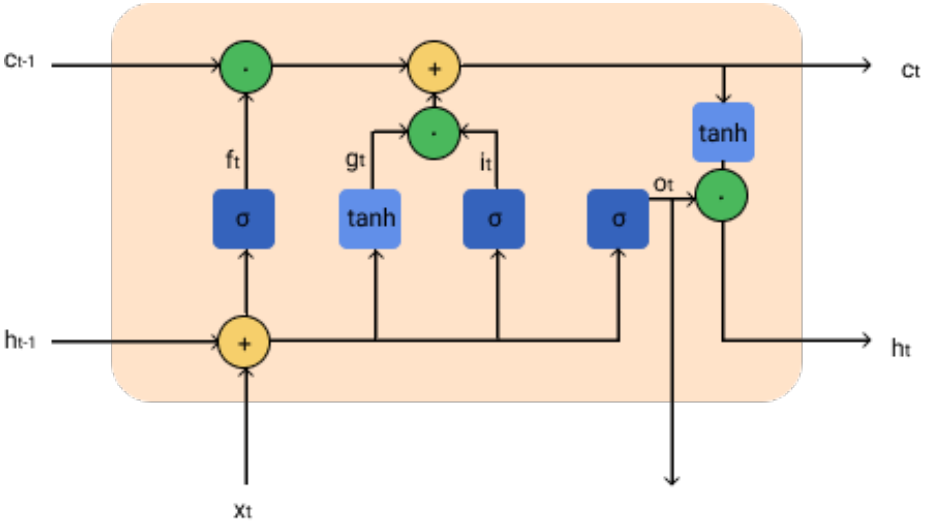

Figure 3.5: LSTM cell model

performance. In this network the interpolation between the hidden state h_t and the previous state h_{t-1} is modulated only by the update gate. If the update gate is close to 1 then GRU uses the new candidate for the hidden state and when it is close to 0 the previous one. They also have a reset gate to calculate the hidden state. There is no cell state.

The hidden state at step t is computed as following. First the reset gate r_t is computed by

$$r_t = \sigma\left(W_{ir}x_t + b_{iz} + W_{hr}h_{(t-1)} + b_{hr}\right), \tag{3.38}$$

the update gate z_t is computed by:

$$z_t = \sigma\left(W_{iz}x_t + b_{iz} + W_{hz}h_{(t-1)} + b_{hz}\right), \tag{3.39}$$

the new gate n_t is defined by:

$$n_t = \tanh(W_{in}x_t + b_{in} + r_t * (W_{hn}h_{(t-1)} + b_{hn})), \tag{3.40}$$

and finally, the hidden state h_t is computed by following:

$$h_t = z_t * h_{(t-1)} + (1 - z_t) * n_t, \tag{3.41}$$

where x is the input, h_{t-1} is the previous hidden state, W_i and W_h, b are learned input weights, recurrent weights and bias.

Every RNN, LSTM and GRU layer can be stacked as multiple layers. In case of NLP these layers can be responsible for different features of language for example one layer is responsible for part of speech and another one learns features responsible for a tense.

3.3.6 CNN

The next architecture is a Convolutional Neural Network [72].It is widely used for an image processing. In natural language processing tasks these networks

combine the information from spatially or temporally local segments. This network is not used in this thesis however it may be used in Natural Language processing and is one of the most fundamental types of Neural Network, so we decided to describe it in this subsection.

The advantages of CNNs is that they provide relatively simple way to detect features of words in a sentence, and they do not suffer so much from vanishing gradient. The disadvantage is that CNN are not natural representation for complicated patterns.

Convolution networks usually contain the layer responsible for convolution operation, activation function and then the pooling operation. There are two kinds of terminology, in the first one three operations are separate layers and in the second the three operations are one layer, in this thesis the second notation is used.

The general convolution operation for two functions f and g is denoted as $*$ and has the form:

$$s(t) = (f * g)(t) = \int_{-\infty}^{\infty} f(t-\tau)g(\tau)\, d\tau \tag{3.42}$$

where f is referred as an input, g is a kernel, t is a time and $s(t)$ is an output at the time t, τ is dummy variable. In machine learning the input is multidimensional array of data. The kernel is multidimensional array of parameters. For discrete input and kernel values the convolution is defined as:

$$s(t) = (f * g)(t) = \sum_{\tau=-\infty}^{\infty} f(t-\tau)g(\tau) \tag{3.43}$$

The output is sometimes called feature map. Often, for image or text processing, there are used two dimensions [53]. An example of equation for two-dimensional input I and two-dimensional kernel K is:

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-n) \tag{3.44}$$

In practice, as convolution is commutative, formula with flipped kernel relative to the input is used more often.

$$S(i,j) = (K * I)(i,j) = \sum_m \sum_n I(i-m, j-n)K(m,n) \tag{3.45}$$

However, most often the cross-correlation is used instead of the convolution:

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(i+m, j+n)K(m,n) \tag{3.46}$$

An example of convolution operation in CNN network is presented in 3.6 The convolution operation is usually followed by an activation function, the most common is ReLU. The next stage is the pooling function. A pooling function takes the output of the previous operation and calculates statistics of nearby outputs. For instance, a max pooling chooses the maximum value among the outputs in a certain area. An average pooling calculates the mean of all outputs in a certain area. The reason for using pooling is to reduce the size of input and make the output invariant to small translations of the input. That means, small changes in positions of input do not change a lot in the pooled output.

After a few convolutional layers there is an output layer, for example, a dense layer that performs classification or regression task.

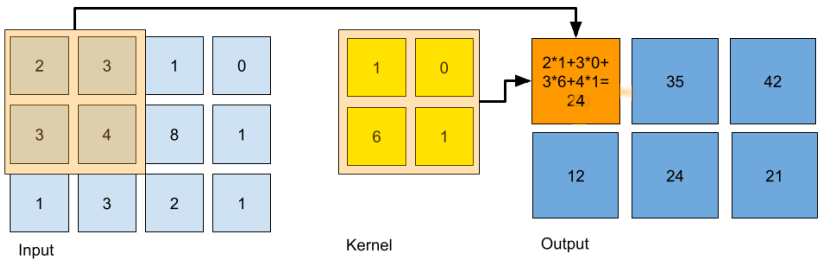

Figure 3.6: Example of convolution operation. Input and kernel are 2-dimensional. Flipping is not used.

3.3.7 Autoencoders

The next type of neural network is an autoencoder [53]. The aim of this network is identity mapping input to the output. This is done using the hidden layer that represents the code for the input. This kind of network has two parts, an encoder $h = f(x)$ that encodes an input x and a decoder that reconstructs the input $r = g(h)$.

The autoencoder is forced to learn which aspects of the data have higher priority. It should not copy the whole input to the output perfectly but only approximate it. For that reason the hidden layer usually has fewer dimensions that the input and output layer. That makes autoencoders learn useful properties. The architecture of this type of network is presented in 3.7.

First autoencoders were used for dimension reduction or feature representation. Encoder and decoder layers can be simple feed-forward layers but there are many variants of autoencoders. An example is a stochastic encoder-decoder, where an encoder provides a conditional distribution $p_{encoder}(h|x)$ and a decoder has to provide a conditional distribution $p_{decoder}(x|h)$ or a denoising autoencoder, where the input has some noise and the aim is to predict the original input.

They can be also used for information retrieval, anomaly detection or machine translation. Encoder-Decoder, a special case of autoencoders used in machine translation, produce an input and output of different lengths.

These models [71] are focused on calculating the probability of $P(E|F)$ of the output sequence E given the input sequence F. The first encoders-decoders use RNNs to encode and decode a sequence.

A sequence $x = (x_1, x_2, \ldots, x_{T_x})$ is an input of length T_x for RNN encoder that calculates:

$$h_t = f(x_t, h_{t-1}) \tag{3.47}$$

and then:

$$c = g(\{h_1, \ldots, h_{t_x}\}), \tag{3.48}$$

where h_t is hidden state at time step t, c is a context vector generated from the hidden states, f and g are non-linear functions.

The decoder has to predict the next word y_t, based on the all previous predictions of words $\{y_1, \ldots, y_{t'-1}\}$ and the context vector c. It is calculated as

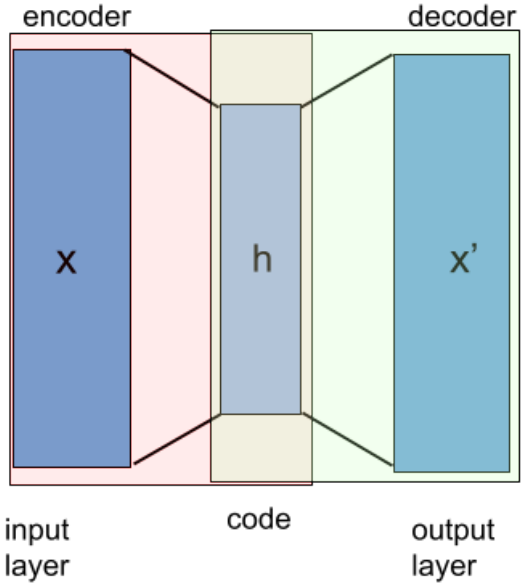

Figure 3.7: Autoencoder's architecture. X is input encoded as h by the encoder and X' is reconstructed input from h by the decoder.

ordered conditional probabilities:

$$p(y) = \Pi_{t=1}^{T} p(y_t|\{y_1, \ldots, y_{t-1}\}, c)) \tag{3.49}$$

where $y = (y_1, \ldots, y_{T_y})$ is the output sequence. Decoder's RNN models each conditional probability as

$$p(y_t|\{y_1, \ldots, y_{t-1}\}, c)) = q(y_{t-1}, s_t, c), \tag{3.50}$$

where q is a nonlinear function, s_t is the hidden state of the RNN.

Nowadays, autoencoders for NLP tasks are replaced by transformers that share some similar concepts and are described in the next subsections.

3.3.8 Attention

For the first time, attention was proposed in a paper [73] in encoder-decoder neural network for neural machine translation. Later, the concept of attention started to be used in other tasks of natural language processing.

The general idea of this model is to search for a set of positions in a source sentence to find the most relevant information, each time the new word is generated. These source positions determine the context vectors that model takes into account together with previously predicted words, when the model predicts a target word.

In this model decoder computes the conditional probability by:

$$p(y_i|\{y_1, \ldots, y_{i-1}\}, c)) = q(y_{i-1}, s_i, c_i), \tag{3.51}$$

where s_i is a hidden state of RNN at time step i computed by:

$$s_i = f(_{s-i}, y_{i-1}, c_i) \tag{3.52}$$

In this equation the difference from 3.50 is c_i. Now each target word y_i is conditioned on a distinct context vector c_i.

The context vector is computed as:

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j, \tag{3.53}$$

where h_j is an annotation from a sequence of annotations $(h_1, \ldots, h_{T_x})$ to which encoder maps the input sequence an α_{ij} is a weight of annotation h_j computed by softmax:

$$\alpha_{ij} = \frac{exp(e_{ij})}{\sum_{k=1}^{T_x} exp(e_{ik})}, \tag{3.54}$$

where e_{ij} is an alignment model that scores the match of inputs around position j and the output at position i:

$$e_{ij} = a(s_{i-1}, h_j) \tag{3.55}$$

The alignment model a is a feedforward neural network that is trained jointly with other components. The decoder decides to which part of an input sentence it should pay attention to, what is called attention mechanism.

Annotations for decoder summarize words based not only on the previous words but also the surrounding context which may include also the following words. For that reason the encoder uses bidirectional RNN which consist of two RNNs: forward and backward. The forward one processes the input sequence in its order and produces a sequence of forward hidden states and the backward - reversed. Hidden states of this two RNNs are concatenated to obtain the annotation for each word. These sequences of annotations are used by decoder in 3.53 and 3.54. The idea of attention mechanism is presented in 3.8

In the above example, described attention is referred as additive attention. There are many variants of attention. In the next subsection the most relevant to this work attention types are described.

3.3.8.1 Self-attention

Attention may be used within a single sentence and then is called self attention [74]. In the basic form, words in an input sequence of length T are represented by word embeddings $X = (x_1, \ldots, x_T)$. The context vector c_i for i^{th} word is:

$$c_i = \sum_{j=1}^{T} \alpha_{ij} x_j, \tag{3.56}$$

where attention weights α_{ij} are computed using a softmax function as in the additive attention:

$$\alpha_{ij} = \frac{exp(\omega_{ij})}{\sum_{k=1}^{T_x} exp(\omega_{ik})}, \tag{3.57}$$

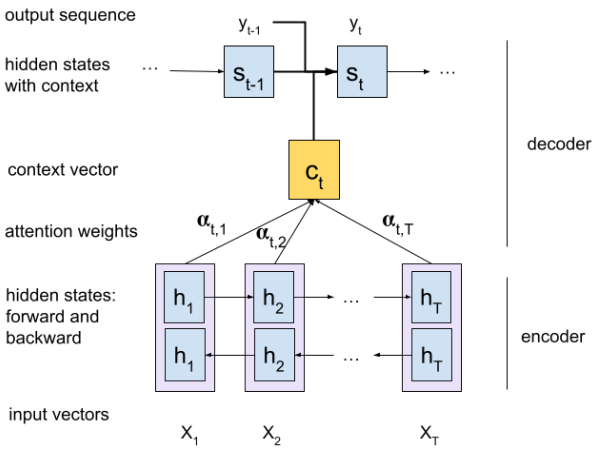

Figure 3.8: Autoencoder with additive attention

.

This time attention weights involve ω_{ij} which is s dot product of two vectors from the input sequence:

$$\omega_{ij} = x_i^t x_j \tag{3.58}$$

The difference with additive attention is ω_{ij} that pays attention to the relation between words in a given sequence rather than words in input and output sequence. This type of attention is called dot-product attention [75]. Dot-product attention is faster than first additive attention [73] which uses a feed-forward network to calculate the attention alignment score.

3.3.8.2 Scaled dot product attention

Before we define the next type of attention we have to explain the concepts of key K, query Q and value V matrices. The mechanism of attention works as a function that maps vectors of a query and set of key-value pairs to an output [73].

The exact meaning of query, values and keys depends on the performed task. For machine translation it may be vectors of input words for queries and output words for keys and values. For language models if self-attention is used the key, values and queries are equal and may be interpreted as vectors of words in a sentence.

In previous subsection self-attention, we calculate a dot product 3.58 of the input vectors x_i, that represent keys k_i, and input vectors x_j, that represent queries q_i. Values are represented by x_j from attention weights 3.57.

Using that notation and using matrix notation for vectors of key K, query Q and values V, we can write the formula for dot product attention's output:

$$Attention(Q, K, V) = softmax(QK^T)V \tag{3.59}$$

An example of a scale dot product attention is presented in figure 3.9. In

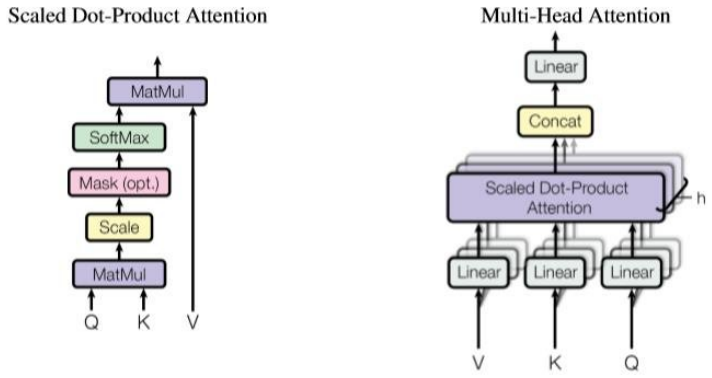

Figure 3.9: Two architectures of attention mechanism. On the left scaled Dot-Product attention which is a part of Multihead Attention on the right. These attentions are used in transformers models. Source of this figure: [76]

this form of attention the matrix of output is computed as:

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V. \tag{3.60}$$

It is similar to the dot product attention except of scaling queries and keys by the factor $\frac{1}{\sqrt{d_k}}$ and d_k is the dimensionality of input queries and keys. Values have dimensiond_v.

3.3.8.3 Multi-head attention

The architecture of this model is presented on the right in figure 3.9.

The model linearly projects queries, keys and values h times with different linear projections to d_k, d_k and d_v respectively. Then the attention function is applied on each of these projections parallel to calculate heads of attention model:

$$head_i = Attention(QW_i^Q, KW_i^K, VW_i^K), \tag{3.61}$$

where the projections are matrices of parameters $W_i^Q \in \mathbb{R}^{d_{model} \times d_k}$, $W_i^K \in \mathbb{R}^{d_{model} \times d_k}$, $W_i^V \in \mathbb{R}^{d_{model} \times d_v}$.

Then, heads are concatenated and projected again to calculate Multi-Head Attention:

$$MultiHead(Q, K, V) = Concat(head_i, \ldots, head_h)W^O, \tag{3.62}$$

where $W^O \in \mathbb{R}^{d_{hd_v} \times d_{model}}$ is also a parameter matrix.

3.3.9 Transformers

Transformer [76] is a language model that may be used for various NLP tasks. The architecture of this model is presented in figure 3.10. It is a sequence-to-sequence model based on encoder-decoder architecture. It relays completely on attention mechanisms and does not use any recurrent or convolution layer.

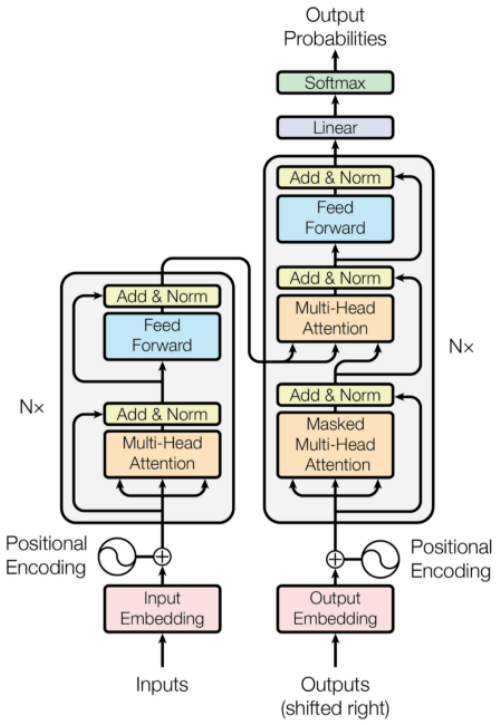

Figure 1: The Transformer - model architecture.

Figure 3.10: Transformer model architecture. The left part is responsible for encoding the sequence and the right part for decoding. Multi-head attention is used to catch the most important elements in the sequences. Source of this figure : [76]

The aim of the encoder in this model is to map input sequence $(x_1, \ldots, x_n)$ to another sequence of continuous representation z $=(z_1, \ldots, z_n)$. Then the decoder is given the vector z and generates the sequence $(y_1, \ldots y_m)$. This model is autoregressive, what means the symbols previously generated, are additional input for the next steps. The illustration of model architecture is presented on 3.10.

The model starts with input tokens that are converted to vectors of dimension d_{model} using learned embeddings. Also output tokens are converted to vectors in the same way. Positional encodings are added to the input tokens to store the information about the order of sequence.

The encoder part is on the left in the picture. It has N stacked identical layers and each layer has two sublayers. The first of two sublayers is a multi-head-attention mechanism. The second sublayer is a fully connected feed-forward network. There is a residual connection around each sublayer. It is followed by a normalization layer. That is, after each sublayer the input and output of this sublayer is summed and normalized. All sublayers and embedding layers in the model have the output of the same dimension equal to 512.

Decoder part is on the right in the picture. It has also N stacked identical

layers. Decoder has two sub-layers similar to encoder. In addition, it has sublayer with multi-head attention over the output of the encoder stack (in the middle in the right picture). Each sublayer is surrounded by a residual connection followed by normalization layer as in encoder. A masked multi-head attention is applied on the output embeddings with position encoding to ensure that predictions at the current position depend only on known outputs from the previous positions. The output of the decoder is converted using linear transformation and softmax function to calculate next-token probabilities. There is the same weight matrix shared between input and output embedding layers and linear transformation after the decoder. The difference is that in the embedding layers the weights are multiplied by $\sqrt{d_{model}}$.

3.3.10 Bidirectional Encoder Representations from Transformers (BERT)

This Bidirectional Encoder Representations from Transformers (BERT) [77] is the model that outperformed existing performance scores on all tasks that it was used for. BERT's aim is to create word embeddings from a textual data. It is similar task as the Word2Vec does but BERT takes into account the context when learning vector representation of words [78]. The difference between left-to-right language models and BERT is that the latter fuse the left and the right context, which stands for bidirectional Transformer. The whole architecture is based on implementation of the encoder from [76] described in the previous paragraphs. The reason for using only encoder part is that we wish to produce words' representation vectors, not to translate sequence into another sequence.

Training BERT involves two steps: pre-training and fine-tuning. BERT is pre-trained using unlabelled text data and different pre-training tasks: masked language model and next sentence prediction. Then, one additional layer is added to initialized during pre-traing BERT to fine-tune this model for a specific task. All parameters are fine-tuned with downstream task like a sentiment analysis, the next sentence prediction or a question answering. Each task has a different model with a dedicated output layer even if the pre-trained model is the same.

There are many freely available pre-trained models of BERT in various languages. In the first paper authors describe two models of varying size: $BERT_{BASE}$ and $BERT_{LARGE}$. The first one has number of layers L=12, for each hidden size H=768 and self-attention heads A= 12. The second, $BERT_{LARGE}$ has a number of layers L=24 , a hidden size H= 1024 and a self-attention heads' number A=16.

BERT has specific input/output representation. It is able to unambiguously represent one or two sentences in one input token sequence. Thanks to that it is able to handle tasks like question answering or next sentence prediction where sentences create some natural pairs.

Sentence is split into tokens based on a token vocabulary. The WordPiece embedding model [79] is used for that. It splits words into sub-words in deterministic data-driven approach. Every sub-word is a separate token. Among tokens for the input and output representation, there are two special tokens: [CLS] and [SEP]. The first one is always added as the first token of a sequence. For this token the final hidden state is used as a representation of a whole sequence for the classification tasks. The second special token is used when two

sentences are packed together into a one sequence, and we wish to separate them by this token. In that case, learned segment embedding added to every token, marks if the token belongs to the first or the second sentence.

Final input representation of a token is a sum of position embedding, segment embedding and token embedding.

After tokenizing, the input sequence is ready for pre-training. First the mask language model task is performed. In this task, the bidirectional representation is trained by masking some percentage of input tokens at random and predicting only those masked tokens. This solution overcomes a problem of standard conditional models trained left-to-right or right-to-left that can't use the whole context of a sequence. Simply using bidirectional conditioning would allow deep model to "see" each target word in a multilayer context before it is predicted. 15% of input tokens in each sentence are selected to be masked. The problem is that [MASK] token does not appear during fine-tuning. Because of that, if the token is selected, then only 80% of time it is replaced by [MASK], 10% of time it is replaced by another random token and 10% of time it is not changed. Then the hidden vector of the replacing input token is used to predict the original token.

Next, sentence prediction task is used to train BERT because many NLP tasks are based on the relationship between two sentences like question answering or natural language inference. This is not captured by previous language models. In order to train this model binary classification task was performed. Two consecutive sentences were chosen from the monolingual corpus. During training 50% of time the second sentence was the actual successor of the first one with label $isNext$ and 50% was a random sentence with label $NotNext$. Token [CLS] is used as a representation of both sentences for next sentence prediction.

Next, BERT model is fine tuned for a specific task. In this step, the proper input and output labels are fed into BERT and end-to-end fine-tuning is done by adjusting pretrained parameters. The input data is specific for task and is tokenized similarly as in pre-trainng. The output layers and acivation functions depend on the task. Fine-tuninig step is usually inexpensive in terms of time and space complexity comparing to pre-training.

Since BERT and transformers succeeded in many NLP tasks transformers become popular and transformed into many new models. Some of them are: RoBERTa [80] that uses more efficient training and smaller number of elements in a network architecture, GPT-2 [81] and newer GPT-3 [82] that are able to generate human-like texts, Electra [83] that uses two transformer models: the generator and discriminator, XLNet [84] that learns bidirectional context and overcomes BERT limitations by autoregressive formulation, DistilBert [85] that learns from pretrained BERT, XLM-Roberta [86] that combines Roberta and XLNet.

Part II

Experiments

Chapter 4

Visibility and Agenda bias

In this chapter we describe helper techniques and tools for news media bias analysis that concern visibility and agenda bias.

In this chapter first we present entities' timeline analysis to reveal visibility bias. Next we propose methods for finding news articles that describe similar events. Finally, we add experiments that show how easy it is to recognize the source of the news.

4.1 Entity timelines experiments

The availability of textual data in web portals gives many opportunities of information extraction. News media create articles about politics and events which should reflect the real world. The amount of information makes it difficult for a human to know and identify all important and objective information. We found it important to extract information from web portals such as news articles and create methods that help to identify most important events.

Analysis of the time series based of news articles or user generated content can give us answers for questions about real life events. We may find out what caused stock market crashes or drop of support of a candidate in president elections. Comparing news texts and user sentiments we can find what topics were the most important in an election campaign.

In our research, we wish to show how timeline of entities' frequency can reflect occurrences of real events and can be used to compare occurrences of entities in different web portals. This section presents research on how to use named entity occurrences in news articles and comments in order to analyse real-world events, their presentation in the media and reception by the on-line users. We present proof-of-concept demo on gathering text from news media, using basic summaries and visualisation of entities that result in promising results.

4.1.1 Problem Specification

The goal of this research is an analysis of entities appearing in political news articles published on leading on-line Polish web portals.

More precisely, the goal is to find if there are any patterns related to named entities encountered in the news articles in time. A named entity is any object

or thing that exists itself and can be named. It can be person, organisation, place, events etc. We wish to make a visual analysis of differences in entity distributions among different time periods.

Additionally, we wish to identify visibility bias. More precisely, we wish to check if there are differences between news portals in a number of mentions of certain entities in their articles. We also wish to analyse co-occurrences of entities.

4.1.2 Experiments

Our approach is to crawl the data from 2 popular web portals and find all entities appearing in each article. We choose portals that are known as representing diverse political views. We recognise named entities in each article and present an analysis of them.

4.1.2.1 Data

A simple crawler developed in Python programming language was used to collect data concerning articles and comments. The language of the textual data is Polish.

We collected the following data: entities in articles, dates of articles, comments to articles, authors of comments and dates of comments.

Web news portals that we consider are the following:

- `wpolityce.pl`
  - 5519 articles
  - 255427 comments
- `gazeta.pl`
  - 521 articles
  - 83832 comments

All the analysed articles are in the category "*politics*". One of the analysed web portals is commonly known as an example of moderate conservative and another as liberal. We have collected data from 01.01.2017 to 22.04.2017.

4.1.2.2 Entity detection

Named entity recognition was done using tool `liner2` [87]. This tool is available by REST API and is a part of Clarin-PL project. This tool recognises named entities and assigns categories to each of them. There are several category models available for `liner2`, but we have noticed that using 4 models increases chances of finding all entities. These models are:

- names - recognizing borders of named entities boundaries
- 5nam - recognizing 5 categories: first name, last name, country, city, street
- top9 - recognising 9 categories: adjective, event, facility, living, location, numex, organization, other and product

- n82 - recognising 9 categories as in top9 and their 82 subcategories.

We excluded two models for temporal entities: timex1,timex4, as we do not use them in our research. We found that sometimes phrases recognised as entities do not represent true entities or sometimes entities are merged when they appear one by one in a text. Despite this, we are able to find the most frequently occurring entities.

We collected entities appearing in the text of articles and article titles. Because of declension of words in Polish, after recognising an entity we take its base form to count its occurrences. For simplicity, we do not focus on entity linking and co-reference resolution i.e. we assume that two different base form entities are always treated separately even if they have the same disambiguation meaning. For example, we collected separate entities 'Tusk' and 'Donald Tusk' even if they most probably represent the same person (a former prime minister of Poland). Despite this, the number of entities appearing as most frequent was sufficiently representative to observe a relation between real world events and higher occurrence frequencies of entities. We are aware about of more advanced NER techniques, however this is out of the scope of this thesis.

We have recognised 24207 unique entities in `wpolityce.pl` articles and 3542 unique entities in `gazeta.pl` articles.

4.1.2.3 Analysis of entities occurrence

We analysed the frequency of entities in news portals. Firstly, we compared data from gazeta.pl and wpolityce.pl and entities' occurrences among different months. Entity frequency is defined as the number of occurrences of each recognised entity in all documents in a month. As a next step, we visualise frequency of entities in time using area plots. This time in visualisation, we use frequency of entities appearing in all articles from one day.

We also analyse co-occurrence of entities in articles from the whole period of time of data we have collected. We can use it to investigate which entities relate to each other. Based on data representing entities in articles we created a matrix, elements of which represent the number of articles where each pair of entities co-occurs.

For example if entity 'Tusk' appears twice and entity 'Kaczyński' (the president of "PiS" ("Law and Justice") the major party in the Polish parliament) appears three times in one article we count it as one co-occurrence. The number of the articles where entity 'Kaczyński' and 'Tusk' appears together is the number of their co-occurrences.

4.1.3 Experimental results of entity occurrences analysis

As the results of our experiments we study the following characteristics:

- the most frequently occurring entities
- entity timelines, i.e. area plots representing occurrence intensity of entities over time
- co-occurrence statistics of entities

January		February	
entity	freq.	entity	freq.
Sejm	315	PiS	132
PiS	242	MON	95
Polska	141	Polska	89
Kaczyński	96	Kaczyński	85
Sala kolumnowa	93	Beata Szydło	75
Jaroslaw Kaczyński	69	Sejm	60
Beata Szydło	51	Warszawa	58
KOD	45	Szydło	54
Macierewicz	44	BOR	53
senat	44	Macierewicz	53
March		April	
entity	freq.	entity	freq.
Polska	247	PiS	160
PiS	225	Polska	56
Donald Tusk	177	Macierewicz	44
Tusk	163	MON	44
Rada Europejska	142	Sejm	38
UE	105	Jaroslaw Kaczyński	35
Sejm	98	PO	30
Kaczyński	93	Donald Tusk	29
polski	88	Warszawa	29
MON	75	Antoni Macierewicz	23

Table 4.1: Most common entities in web portal gazeta.pl

The results presented in this chapter concern some example entities to illustrate our approach.

Tables 4.1 and 4.2 present the most frequent entities among the first 4 months in 2017. Some entities are the most frequent ones in both portals other appears just in one of them. We can observe that some entities are encountered with similar frequency in every month. These are entities which give us less information about current popular topics in the news. Not surprisingly for each month for both portals 'Polska' (Poland) entity is among the most common entities. Similar popularity is observed for the entity 'PiS' the name of major party in the Polish parliament.

One can notice that some entities appear more often in one web portal than in another. We observed the following phenomenon: in the first month of analysis, gazeta.pl more often uses entities related to the currently ruling party, when wpolityce.pl represents the higher frequency of entities related to opposition.

Figures 4.1 and 4.2 present intensity timelines of three example entities from the most frequent ones: 'Sala kolumnowa' ('Column Hall') - the name of the room in the Polish parliament where the voting on the Polish budget had to be exceptionally moved in December 2016, 'MON' (abbreviation of the name of the Polish Ministry of Defense) and 'Donald Tusk' (a former prime minister of Poland, who resigned in 2014 for taking a position of a president of the European council). It can be noticed that entity distributions in these examples seem to be strongly associated with some real events of high importance in Polish politics

January		February	
entity	freq.	entity	freq.
Polska	2712	Polska	2383
Sejm	2293	PiS	1466
PiS	1927	Warszawa	1092
KOD	895	polski	645
Polak	758	UE	638
Kijowski	594	Sejm	532
Nowoczesna	581	Polak	499
zloty	537	Niemcy	496
polski	537	zloty	493
TK	515	Europa	481
March		April	
entity	freq.	entity	freq.
Polska	3551	Polska	1264
PiS	1818	PiS	1210
UE	1417	Polak	519
Donald Tusk	1194	Donald Tusk	438
Tusk	1088	Tusk	429
polski	1015	Sejm	390
Europa	996	MON	373
Polak	801	zloty	332
Rada Europejska	722	polski	311
Unia Europejska	629	UE	293

Table 4.2: Most common entities in web portal wpolityce.pl

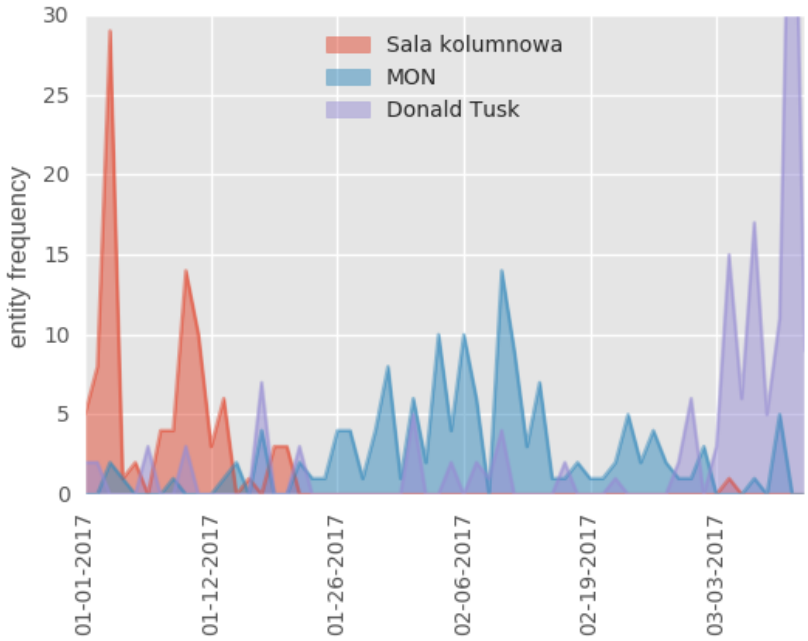

Figure 4.1: Example of timeline for entities: Sala Kolumnowa, MON, Donald Tusk encountered in news articles at gazeta.pl from 01.01.2017 to 10.03.2017

that actually took place at that time.

First of all, we can observe a high frequency of the entity 'sala kolumnowa' at the beginning of the year. 'Sala kolumnowa'. The timeline shows that this place was often mentioned in Polish news in January. After some time this entity is almost not mentioned in articles of both portals anymore. With a background knowledge of the political situation in Poland, we can draw a suggestion that this represents particular event: Sejm meeting unexpectedly took place in Column Hall due to the illegal blocking of the main voting room by the opposition. This event provokes discussion about the legality of the blocking or validity of the voting between the ruling party and the opposition what had a reflection in the news from that time.

Also for entity 'Donald Tusk' which represents Polish politician, former prime minister, we can find some interesting pattern. Just before a day of his election for President of the European Council the number of occurrences rapidly grows for both portals. Again it was a controversial event in the Polish political scene.

The third entity 'MON' is related to the Polish Ministry of National Defence. Higher frequency of this entity observed in February 2017 seems to be naturally related to an affair that actually took place at this institution that time.

Except for mentioned similarities, we can observe some differences. For example 'MON' entity is mentioned rather regularly at portal 'wpolityce.pl', and at 'gazeta.pl' mainly at February. 'Donald Tusk' entity occurs more often at 'wpolityce.pl' portal than in 'gazeta.pl' in the first two months. Entity 'MON' is more popular than entity 'Donald Tusk' at 'gazeta.pl' but 'Donald Tusk' entity

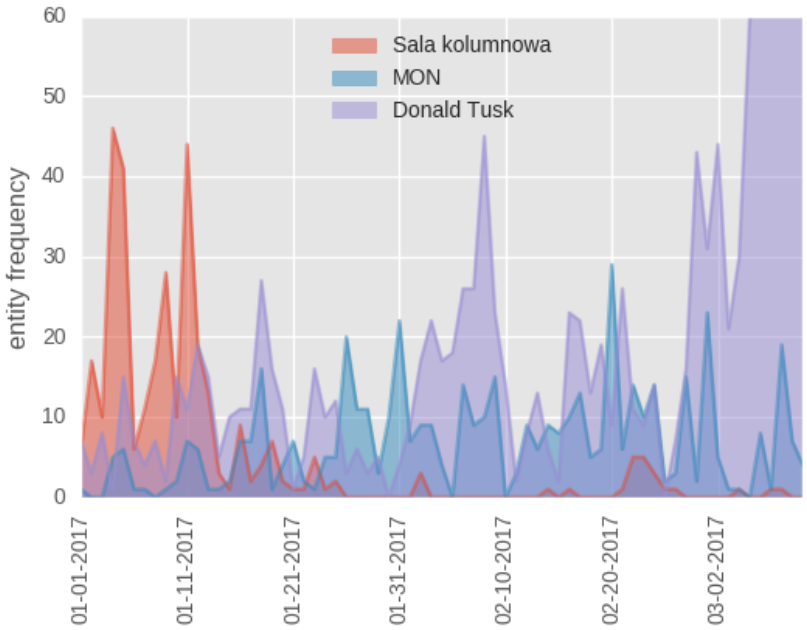

Figure 4.2: Example of timeline for entities: Sala Kolumnowa, MON, Donald Tusk encountered in news articles at wpolityce.pl from 01.01.2017 to 10.03.2017

is, in general, more popular entity than 'MON' at 'wpolityce.pl'.

Co-occurrence of entities to exemplary entities are shown in table 4.3 for portal gazeta.pl and table 4.4 for wpolityce.pl. Each examplary entity has its co-occurring entities. Entities are considered as co-occuring if they encounter in the same article at least once. In each case entities 'Polska', 'PiS' are repeated because they are among most frequent ones and relate to the country and government. For entity 'Donald Tusk' the most interesting one is 'Rada Europejska' (EU Council) which is strongly connected to his current function. 'MON' occurs most often with entity 'Macierewicz' and 'Antoni Macierewicz' which relates to the person of the current Minister of National Defense. For entity 'Sala kolumnowa' there is some difference between both portals. Both use entities 'sejm' (parliament), 'PiS' (the ruling party), 'Polska' but only in one the phrase 'Sala kolumnowa' is mentioned with 'PO Michał Szczerba' (a politician from the current opposition known for initialising the illegal process of blocking the work of the Polish parliament what resulted in moving the work into the 'Sala kolumnowa' room) and 'PO i Nowoczesna' (the names of the current opposition parties in Poland) but at 'wpolityce.pl' it co-occurs with 'senat' (the higher chamber of the Polish parliament).

4.2 News similarity

One of basic modules in any bias-analysis tool is a module that makes it possible to automatically or semi-automatically detect *pairs* of news articles (or, more generally: text documents), that report on the same event, topic or entity.

Donald Tusk		MON		Sala kolumnowa	
entity	freq.	entity	freq.	entity	freq.
Rada Europejska	66	Macierewicz	58	sejm	30
PiS	64	Antoni Macierewicz	50	PiS	21
Tusk	63	PiS	31	PO i Nowoczesna	16
Polska	63	Polska	29	PO Michał Szczerba	13
polski	42	NATO	26	PO	12

Table 4.3: Co-occurring entities in web portal gazeta.pl

Donald Tusk		MON		Sala kolumnowa	
entity	freq.	entity	freq.	entity	freq.
Polska	729	Polska	249	sejm	134
Tusk	606	Antoni Macierewicz	206	PiS	95
PiS	526	Macierewicz	199	Polska	72
Rada Europejska	463	PiS	177	PO	52
polski	155	polski	155	senat	46

Table 4.4: Co-occurring entities in web portal wpolityce.pl

Such pairs of articles, where each article comes from a different *source* (e.g. web portal, particular author, etc.) can be further used to make comparison-based analysis towards detecting information bias. The pairs are also necessary to build a training, test or reference set in the case of machine-learning approach to the described research problem.

In this subsection we present a method and experimental results of detecting pairs of news articles on the same (similar) topic or reporting the same (similar) event, etc. We focus here on the news articles in news portals.

4.2.1 Problem Specification

In this section we consider two research problems:

- news similarity detection problem
- information source recognition problem

4.2.2 News similarity detection

We proposed two approaches to news similarity detection: find all similar articles to the given one and given two articles decide if they are similar or not. The first approach to the problem is as follows. In a given collection of news articles from a given time window (e.g. particular day, etc.) detect the groups of articles that report on the same topic/event, etc. A manually labelled training set that is a collection of manually grouped news articles is prepared. We apply text mining techniques to identify the similar events. The models are evaluated using the metrics presented in the next section: averaged precision, averaged recall and averaged F-measure.

The second approach is to identify whether 2 articles are similar. We apply the machine-learning approach to this problem. For each article there are computed several attributes based on the textual contents, keywords, etc. Then,

news portal	number of articles
gazeta.pl	2623
dorzeczy.pl	4197

Table 4.5: Dataset

the set is used to train some ML models. Finally, the models are used to automatically detect similar articles. The models are evaluated using the prepared group labels and some standard measures such as precision, recall or f-measure.

4.2.3 Towards news bias detection

The second research problem studied in this section is the following. Given an article and the set of information sources (e.g. web news portals) is it possible to automatically recognise which source does this article comes from based only on the contents? This kind of experiment can be viewed as a one of simple tests of imparity of information sources. I.e. if it is possible to correctly predict the source of the news article based on its content then it is more likely that this information source has some information bias.

Of course some other reasons may make it possible to predict the source of the news article including the writing style, etc. However this simple test may serve as one of the multiple tools that could in ensemble help to detect information bias. More advanced bias-detection tools are envisaged in our ongoing research.

4.2.4 Experimental Setup

4.2.4.1 Data Collection

We collect the data from two Polish news portals: 'dorzeczy.pl' and 'gazeta.pl'. These two are chosen from among the most popular Polish news portals. In addition, they are considered by many readers as examples of media having completely different views on the reality in Poland especially in the domain of social issues or politics and hence making it possible to build an interesting dataset with a potential of containing pairs (or clusters) of articles on the same/similar topic/event/entity but with potentially various forms of information bias. Articles are categorised by a predefined set of topic categories on each of the portals. The decision was made to focus on events connected to the politics in Poland or world. In this case we were looking for articles from category 'Information' ('Wiadomosci') in 'gazeta.pl' and in portal 'dorzeczy.pl' for categories 'Country' and 'World' ('Kraj' and 'Swiat').

In these experiments we decided to focus only on Polish media but it is possible to extend our research to other languages.

We have collected the articles from 01.01.2018 to 07.04.2018. Table 4.5 presents the number of articles.

4.2.4.2 Database

We store the data in MongoDB - a document database. We create article collection of news articles and their comments. Each item in the collection represents

group quantity	number of groups
1	145
2	36
3	17
4 or more	16

Table 4.6: Distribution of articles among groups

an article and contains the following fields: _id', 'article_id', title', 'date', 'lead_text', 'text', 'keywords', 'source', 'url' and 'comments'. Field comments contains 'author', date', 'comment_id', 'text'.

4.2.4.3 Data Annotation

For news similarity detection we needed to manually create an annotated data set. The common approach for text similarity recognition is to create a set of article pairs and annotate if they are similar or not. We realised that for news articles this approach may not be the best one. We wish to find all articles that are similar and sometimes one news portal describes an event in one article and other news portal writes about this in a series of four articles, for example. Thus we define the task of annotating similar articles as follows.

For a given time window (e.g. a particular date) we collect all articles from the specified web news portals. Each article is assigned to a group with articles about similar event. If there is no group with articles describing the event create a new one. The group contains all articles about the same event.

We have annotated 385 articles from 6 randomly chosen days. Articles formed 213 groups. There are groups of consisting of one article or groups containing many articles. The distribution of articles among groups quantity is presented in table 4.6 Each record of annotated data contains (among others) the following attributes: 'date', 'article_id', 'group_id'.

4.2.4.4 Data Preprocessing

In the preprocessing phase we apply several operations including: removing stop-words, normalising - convert words to the base form using `Morfeusz` library[1].

4.2.4.5 Evaluation Measures

In order to evaluate experimental results we calculate the average precision, recall and f measure in each experiment.

The average precision is the average of precision of each group. That is given by the following expression:

$$ap = \frac{1}{N}\sum_{n=1}^{N} p_n = \frac{1}{N}\sum_{n=1}^{N} \frac{tp_n}{tp_n + fp_n} \tag{4.1}$$

[1] `http://sgjp.pl/morfeusz/morfeusz-siat.html`

Algorithm	ap	ar	af1
Keywords similar.	0.60	0.57	0.42
Doc2Vec + cos sim	0.72	0.57	0.50
Doc2Vec+bigram+cos simililar.	**0.93**	0.60	0.64
Doc2Vec+trigram+cos similar.	0.92	0.63	0.66
TF-IDF +cos similar.	0.50	**0.69**	**0.53**

Table 4.7: Evaluation of article's similarity detection

Where N is the number of evaluated groups. Accordingly average of recall is given as:

$$ar = \frac{1}{N}\sum_{n=1}^{N} r_n = \frac{1}{N}\sum_{n=1}^{N} \frac{tp_n}{tp_n + fn_n} \tag{4.2}$$

Finally, average F-measure is defined as follows:

$$af1 = \frac{1}{N}\sum_{n=1}^{N} \frac{2 * p_n * r_n}{p_n + r_n} \tag{4.3}$$

4.2.5 Experimental Results on Articles Similarity Detection

4.2.5.1 Group approach

In a group approach of finding similar articles we experimented with three methods for the news similarity detection problem:

- keyword set similarity - this is our simple baseline solution. We compared the number of similar keywords and find the most similar articles using predefined threshold based on the number of keywords.

- tf-idf with cosine similarity- after preprocessing of a textual data, we calculated tf-idf and cosine similarity between articles from a given data frame. Again we choose the most similar articles based on the predefined threshold.

- doc2vec [88] with cosine similarity in three variants: unigrams, bigram phrases, trigram phrases. For Each of these we choose doc2vec based on bag of words model. Similar preprocessing was done as for tf-idf.

The results of evaluation are presented in a table 4.7. The best averaged results for given measures were highlighted in bold. The best average precision is observed for two doc2vec models. That means for these models there are the least false positives. However, the best f-measure and recall is observed for tf-idf algorithm. This algorithm is better choice if we wish to find as many similar articles as possible without caring about dissimilar articles among them.

news portal	training set	test set
similar (1)	320	151
not similar (0)	7837	2624

Table 4.8: Number of article pairs for similarity detection

Algorithm	Class	Precision	Recall	F1-score	Support
Siamese	0.0	0.95	0.86	0.90	2624
LSTM	1.0	0.07	0.19	0.10	151
	avg / total	0.90	0.82	0.86	2775
SVM	0.0	0.98	0.91	0.94	2624
polynomial	1.0	0.29	0.63	0.40	151
kernel	avg / total	0.94	0.90	0.91	2775
SVM	0.0	0.98	0.85	0.91	2624
linear	1.0	0.20	0.68	0.31	151
kernel	avg / total	0.94	0.84	0.88	2775
Logistic	0.0	0.98	0.89	0.93	2624
regression	1.0	0.24	0.63	0.35	151
	avg / total	0.94	0.87	0.90	2775
Gradient	0.0	0.96	0.95	0.96	2624
boosting	1.0	0.27	0.28	0.27	151
classifier	avg / total	0.92	0.92	0.92	2775

Table 4.9: Evaluation of one to one articles pairs. Class "1" means articles are about the same topic and "0" means they are not about the same topic

4.2.5.2 Pair approach

In this task we wished to identify if a pair of articles is similar or not. The data was split into test and train datasets as presented in 4.8. Similar articles was labelled as '1' and not similar articles as '0'. We have created the following features as input vector for each pair: cosine similarity on tf-idf vectors, number of similar keywords, normalized number of similar entities that is number of similar entities/sum of entities in both articles. In table 4.9 we present an evaluation of proposed algorithms.

All algorithms have quite good results. Support vector machines occur to be the best one. Siamese neural networks has high results in total but very low scores for similar pairs where the reason may be that it was not able to detect dependencies in long text.

4.2.6 Experimental Results on News Article Source Detection

We experimented with three machine learning algorithms in the problem stated as prediction of the news article source based on its content. In all the experiments concerning this problem, the articles' attributes explicitly mentioning the actual source (e.g. the "source" attribute) were ignored in the prediction phase. Dataset for this task is presented in table 4.10. We have used the following algorithms: naive bayes, logistic regression, support vector machines.

news portal	training set	test set
gazeta.pl	2436	395
dorzeczy.pl	3591	606

Table 4.10: Number of articles for media outlet detection

Table 4.11: Evaluation of article's media outlets detection

algorithm	news portal	precision	recall	f1-score	support
Naive Bayes	dorzeczy.pl	0.69	0.98	0.81	606
	gazeta.pl	0.89	0.32	0.47	395
	avg / total	0.77	0.72	0.67	1001
Logistic Regression	dorzeczy.pl	0.90	0.83	0.86	606
	gazeta.pl	0.76	0.86	0.81	395
	avg / total	0.85	0.84	0.84	1001
SVM	dorzeczy.pl	0.88	0.86	0.87	606
	gazeta.pl	0.79	0.83	0.81	395
	avg / total	**0.85**	**0.85**	**0.85**	1001

The evaluation of proposed methods is presented in table 4.11. Support vector machines have the best score for pairs of articles (dentoed as 1) slightly outperforming logistic regression. These results show that based on simple approach, analysing the basics of used language we are able to recognise the source.

4.3 Summary

In this section we presented helper tools that may be used in the online news media bias detection.

Chapter 5

Entity-based news sentiment analysis

In this chapter the main results of this thesis are presented. We study the entity-level sentiment detection in news headlines. We proposed several models and compared them with state-of-the-art approaches. We achieved similar or better results.

5.1 Motivation

Many ordinary users of the news portals usually quickly and superficially read lots of news headlines every day, often without deeper background knowledge. The news headlines read in this way by the users may influence the way they view the world, often unconsciously. It especially concerns political news media where the stake is potentially high (e.g. supporting by the reader one or another politician in elections, etc.).

It is possible that media outlets often intentionally construct the headlines to present an entity (e.g. a politician) in a positive (or negative) light. Examples are:

"Trump is projecting an image of strength amid battle with coronavirus"

"Trump actually believes he can sell himself to America as a COVID-conquering hero",

to mention two real political news headlines concerning 2020 presidential elections in USA and coming from on-line media and mentioning a major politician in a perhaps non-neutral way.

It is hard to precisely define what exactly makes positive or negative impression (concerning the politician) on the reader while reading such headlines, and it might be partially unconscious for the reader. But the fact that majority of readers of a news headline independently have similar impression concerning the mentioned entity might be a signal that the phenomenon is real.

Obviously, one of the main tools to achieve this effect is biased language. However, even if the language is objective (neutral), the positive or negative entity-level tonality might be achieved for example by selecting the context (facts, other entities, etc.) in which the entity is mentioned.

Dataset	final size	headlines excluded	an-ns	an-rs	R-κ	F-κ	% agr.
SEN-pl	1188	168	4441	19	0.512	0.459	80.89
SEN-en-R	1271	40	3913	19	0.406	0.309	76.22
SEN-en-AMT	1360	55	4503	57	0.396	0.303	75.95

Table 5.1: SEN datasets summary. Column name an-ns is number of annotations and an-rs is number of anntoators.

Thus, providing a human-labelled benchmark dataset for training and testing machine-learning algorithms for sentiment concerning entities in political news headlines seems to be valuable for the research community.

In this section we present a new publicly available benchmark dataset for bi-lingual entity-level sentiment analysis in news headlines. We hope that introducing such data would make it possible to work towards providing tools for supporting more objective and fairer media that would be beneficial for the society. To our best knowledge this is the first labelled dataset concerning this domain that includes Polish.

5.2 The SEN dataset

For our experiments we created a novel bi-lingual benchmark dataset called "SEN" ("Sentiment concerning Entity in News headlines").

The set is publicly available[1] and consists of 3819 human-labelled and curated records. Each record contains a news headline, a named entity mentioned in the headline and a human annotated label (one of *"positive", "neutral", "negative"*[2]).

Our SEN dataset package consists of 2 parts: SEN-en (2631 English headlines that split into SEN-en-R and SEN-en-AMT), and SEN-pl (1188 Polish headlines). Each headline-entity pair was annotated via the open-source annotation tool doccano[3] by at least 3 annotators from a team of volunteer researchers (the whole SEN-pl dataset and a subset of 1271 English records: the SEN-en-R subset, "R" for "researchers") or via the Amazon Mechanical Turk service (a subset of 1360 English records: the SEN-en-AMT subset). Table 5.1 summarises the details.

5.2.1 Collecting and selecting the data

The headlines were crawled from an intentionally diversified set of popular online news media outlets. In total we gathered 212.982 articles in Polish and 136.379 in English. In our collection of articles the earliest news in Polish dataset has a date 03.05.2019 and the latest 16.10.2020. The earliest English collection article is from 19-10-2001 and the latest 14-04-2021.

As some outlets do not provide access to archival articles and some reduce the number of them, to achieve consistency and typical number of titles in each

[1] `https://zenodo.org/record/5211931`

[2] The fourth, special "unknown" label was available only during the annotation process when selecting any other label was problematic for an annotator

[3] https:/github.com/doccano/doccano

English				Polish		
source	total	SEN-en-R	SEN-en-AMT	source	total	SEN-pl
nytimes.com	23913	491	470	niezalezna	36015	140
wsj.com	88489	47	99	onet.pl	44807	139
foxnews.com	4452	260	250	tvn24	31193	64
truepundit.com	12890	121	191	wpolityce	28971	664
washingtonpost.com	6635	392	405	natemat	1313	7
				tvp.info	20750	58
				dorzeczy.pl	18900	112
				polsatnews	15751	58

Table 5.2: The number of downloaded articles (total) and those selected for the annotation

news portal for our research we decided to use the data ranging from December 2019 to February 2020. From this period of time we selected headlines that contain selected named entities. We selected the entities manually to represent some active politicians, famous people, political parties or countries.

We assured to have certain minimum number of headlines about each entity. We selected the entities manually, since some NER tools that we have tested miss entity mentions when they are at the beginning of the sentence, what is a common case in news headlines, and some entities with surnames being common words are not recognized by NER. Statistics of collected data are presented in table 5.2.

The entities selected for the research are:

- for SEN-en: 15 named entities being popular politicians or celebrities, with a special emphasis on the candidates on current US presidential campaign, including: *Trump, Biden, Sanders, Bloomberg, Putin, Thunberg,* etc.
- for SEN-pl: 17 named entities being popular politicians, political parties, etc. including: *Duda* (the current Polish president), *Tusk* (former Polish prime minister), *Trump, Putin, Macron, Thunberg,* etc. and the country name *"Polska" ("Poland").*

Detailed proportion of labels for each entity is presented on figure 5.1

5.2.2 Data preprocessing, cleansing and annotation

The data is annotated by two groups of annotators: voluntary researchers and workers of a crowdsourcing tool Amazon Mechanical Turk (AMT) (only for English).[4]

For the first group of data annotation we created a survey in doccanno [89] framework. Before final annotations we made a few trial annotations with different data annotation rules with 20 headlines each. We analysed the annotators' questions and their performance.

For example, we clarified that the annotators should not evaluate the general sentiment of the whole headline when it does not express any sentiment towards the target entity. We instructed the annotators to not interpret subtle irony or sarcasm. We observed that annotators perform better when the annotation rules are strict and short. For the final version we asked volunteer researchers and academics to annotate the data. Each annotator had access to a few packages

[4]We did not use AMT for Polish language as we did not find relevant number of annotators and the number of annotations was too small.

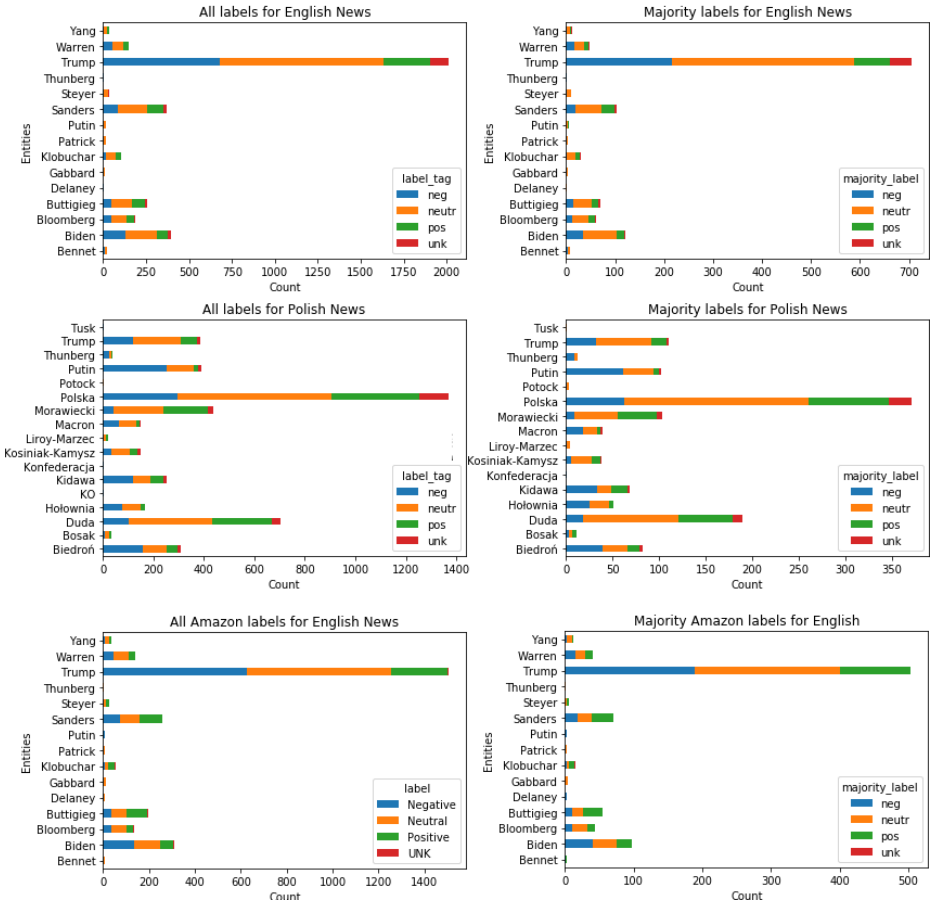

Figure 5.1: On the left: 3 pictures with number of all class labels that were assigned to each entity by anntoators. On the right: final class labels after aggregation.

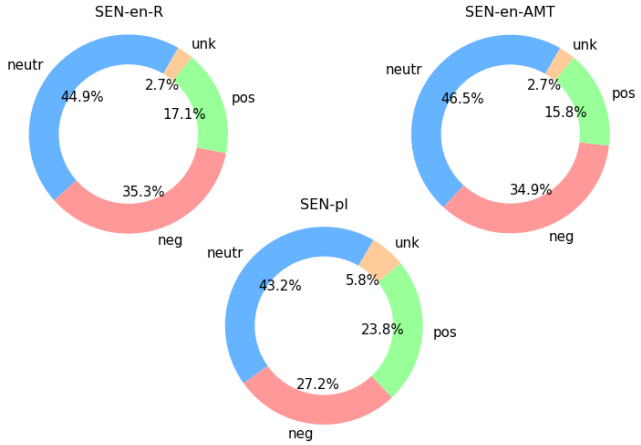

Figure 5.2: Label distribution in the SEN dataset

of data each of 100 headlines. It was recommended to annotate at least one package or more. During the process of annotation it was assured that each headline has at least 3 annotators.

Each annotator was presented a headline-entity pair and was expected to select one of possible lablels: "positive", "neutral", "negative" and "unknown". Similar prerequisites were used for the AMT workers.

The final set of annotation rules was based on our experience from the first phase and on some existing works [32], [31]. Examples of annotation rules presented to the annotators are as follows:

- decide whether the entity is presented in negative, neutral or positive light
- sentiment may be revealed by clear statement or opinion about an entity
- do not focus on the whole headline sentiment but only on the sentiment of the entity, how this entity is described
- in case the sentiment is not possible to be determined use the "unknown" label (it may happen, for example, when the headline strongly depends on the context or the sentiment is mixed, etc.)
- try to be neutral and objective (supress your personal opinion on the entity)

Annotators were expected to use only the first impression just after having read the headline and try to abstract from their subjective personal opinions on the entities or events, etc. The records which obtained ambiguous labels, i.e. 3 different labels, or at least 2 “unknown” labels were excluded from further processing. The final labels were aggregated via majority voting. See Figure 5.2 for label distribution.

5.2.3 Outlier annotator detection

In order to detect outliers among the annotators we computed the Jensen-Shanon entropy divergence (JSD) measure [90] of each annotator and inspected

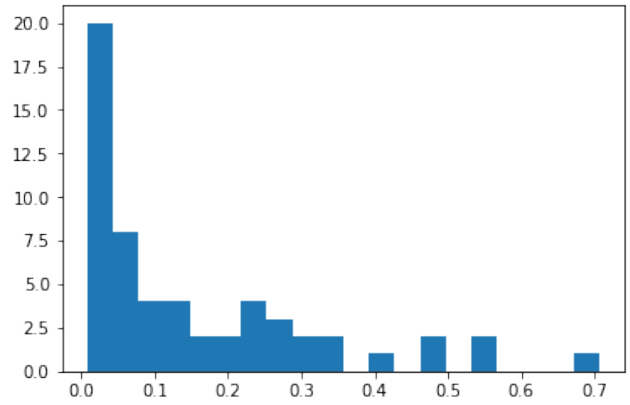

Figure 5.3: SEN-en-AMT annotators Distribution of JSD values

.

the distributions of the JSD value in each of the 3 parts of the SEN dataset.

JSD measure The JSD measure can be viewed as a variant of the Kullback-Leibler divergence and is defined as follows:

$$JSD(p_i||q_{ij}) = \sqrt{\frac{D(p_i||m) + D(q_{ij}||m)}{2}} \tag{5.1}$$

where p_i is the label distribution vector for the i-th entity in the final dataset, q_{ij} is the label distribution vector for the i-th entity and j-th annotator.

To curate the labels, we did not take into the account the labels of the top-JSD value annotators. The figures 5.4 and 5.5 present the distibutions of the JSD values. It can be observed that in all datasets there are some outlier annotators. We excluded the annotators that were more than 2 standard deviations away from the mean of the distributions. There were 6, 4 and 2 such annotators in the sets SEN-en-AMT, SEN-en-R and SEN-pl, respectively (see the top-right bars of the histograms on the figures).

5.2.3.1 Detecting entity-related bias of annotators

Furthermore, we analysed the annotators in terms of their bias towards the annotated entities. To this end we applied the sentiment score measure.

Sentiment score measure The sentiment score measure is defined as follows:

$$Sentiment\_score(Entity) = \frac{pos - neg}{pos + neg + neutr}, \tag{5.2}$$

where pos, neg and neutr is the number of positive, negative or neutral annotations, for the given setting, respectively. The score equal to 1 means maximal positivity, etc.

We have calculated sentiment score for each entity-annotator pair, see figures 5.6, 5.7 and 5.8. The labels of the outlier entity-annotator pairs (represented as the dots over or below the boxplots) were excluded from further considerations.

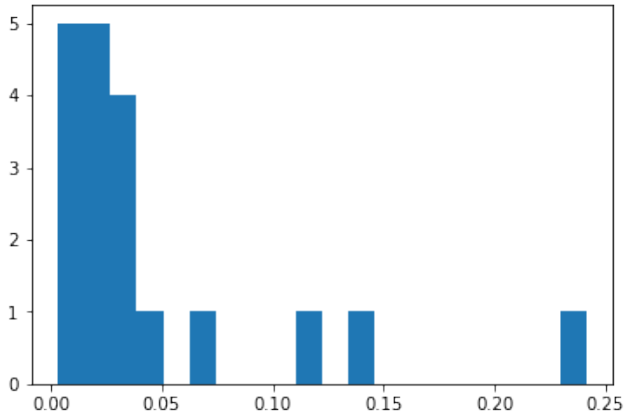

Figure 5.4: SEN-en-R annotators. Distribution of JSD values

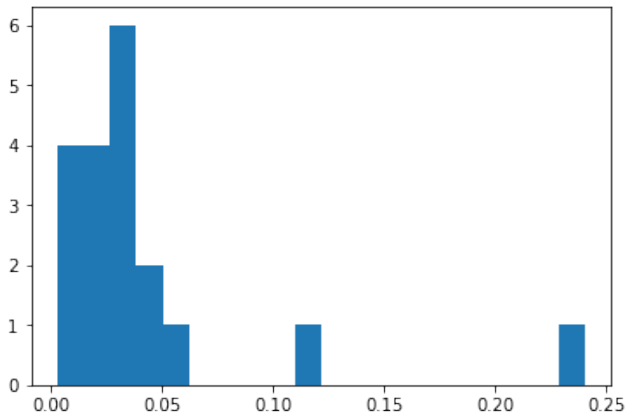

Figure 5.5: SEN-pl-R annotators. Distribution of JSD values

entity	number of annotations	pos	neutr	neg	sentiment score	entropy
Biedroń	307.0	41	94	160	-0.403390	1.400166
Bosak	36.0	9	17	9	0.000000	**1.513697**
Duda	704.0	236	333	101	0.201493	1.443068
Hołownia	168.0	17	69	79	-0.375758	1.372542
KO	3.0	0	0	3	-1.000000	0.000000
Kidawa	251.0	9	69	121	-0.301255	1.483329
Konfederacja	3.0	0	0	3	-1.000000	0.000000
Kosiniak-Kamysz	149.0	32	74	33	-0.007194	1.464520
Liroy-Marzec	23.0	9	9	4	0.227273	1.502220
Macron	149.0	14	69	62	-0.331034	1.359555
Merkel	3.0	0	3	0	0.000000	0.000000
Morawiecki	439.0	175	197	43	0.318072	1.374472
Polska	1369.0	348	610	295	0.042298	1.510150
Potocki	6.0	0	4	1	-0.200000	0.721928
Putin	391.0	15	109	253	-0.631300	1.088838
Rosja	3.0	0	2	1	-0.333333	0.918296
Thunberg	40.0	5	10	24	-0.487179	1.314427
Trump	388.0	68	189	118	-0.133333	1.469778
Tusk	6.0	0	0	6	-1.000000	0.000000
Śpiewak	3.0	1	2	0	**0.333333**	0.918296

Table 5.3: Polish entities statistics and measures. The lowest sentiment score and entropy is underlined. The highest is bold.

5.2.4 General entity bias detection

In addition we applied the sentiment score measure (defined in the subsection 5.2.3.1) to each entity in the dataset. The results are presented in the tables: 5.3, 5.4, 5.5.

We also applied the entropy measure to see whether the entities vary in diversification of sentiment labels.

It can be observed that some entities are strongly biased in the dataset, i.e. most of the headlines that concern them are either positive or negative. Examples are: "Trump", "Biden", "Putin", etc. Because of this entity-bias observation, we decided to use several techniques to achieve better performance of machine-learning algorithms trained on this dataset.

5.3 Experiments on entity-level sentiment classification on the SEN dataset

In this section we present the experiments concerning one of the main forms of bias detection, namely tonality bias detection in headlines of news articles. More precisely the input of the classifier is the headline of the article and the entity (for example famous politician such as Donald Trump, etc.). The output is one of the labels: "positive", "neutral", "negative". It can help to detect whether the author of the article and headline manipulates the reader's reception of the

entity	number of annotations	pos	neutr	neg	sentiment score	entropy
Bennet	24.0	2	11	11	-0.375000	1.330484
Biden	416.0	73	189	140	-0.166667	1.488811
Bloomberg	208.0	51	95	55	-0.019900	1.524660
Buttigieg	271.0	85	126	51	**0.129771**	1.494380
Delaney	2.0	0	1	1	-0.500000	1.000000
Gabbard	9.0	1	5	3	-0.222222	1.351644
Klobuchar	116.0	34	62	18	0.140351	1.418920
Patrick	21.0	0	11	8	-0.421053	0.981941
Putin	18.0	4	11	3	0.055556	1.347223
Sanders	413.0	108	193	97	0.027638	1.513349
Steyer	35.0	5	23	6	-0.029412	1.229776
Thunberg	5.0	1	2	2	-0.200000	1.521928
Trump	2167.0	297	1000	750	-0.221299	1.439697
Warren	160.0	34	67	57	-0.145570	**1.53241**
Yang	48.0	12	29	6	0.127660	1.311806

Table 5.4: SEN-en-R entities statistics and measures. The lowest sentiment score and entropy is underlined. The highest is bold.

entity	number of annotations	pos	neutr	neg	sentiment score	entropy
Bennet	6.0	1	1	4	-0.500000	1.251629
Biden	388.0	62	145	176	-0.297650	1.471261
Bloomberg	167.0	44	74	47	-0.018182	1.543412
Buttigieg	243.0	106	85	50	0.232365	1.522223
Delaney	9.0	0	5	4	-0.444444	0.991076
Democratic Party	33.0	4	8	20	-0.500000	1.298795
Democrats	682.0	121	219	331	-0.312966	1.475775
GOP	183.0	47	51	80	-0.185393	1.542511
Gabbard	11.0	0	10	1	-0.090909	0.439497
Klobuchar	75.0	34	27	11	**0.319444**	1.455905
Patrick	6.0	0	3	3	-0.500000	1.000000
Putin	12.0	2	3	7	-0.416667	1.384432
Republican Party	10.0	4	3	3	0.100000	1.570951
Republicans	294.0	76	102	108	-0.111888	1.569110
Sanders	308.0	112	97	99	0.042208	**1.581976**
Steyer	30.0	10	16	4	0.200000	1.399581
Thunberg	2.0	0	2	0	0.000000	0.000000
Trump	1837.0	287	738	804	-0.282668	1.468850
Warren	168.0	34	78	55	-0.125749	1.508196
Yang	39.0	9	21	9	0.000000	1.457266

Table 5.5: SEN-en-AMT entities statistics and measures. The lowest in the column sentiment score and entropy is underlined. The highest is bold.

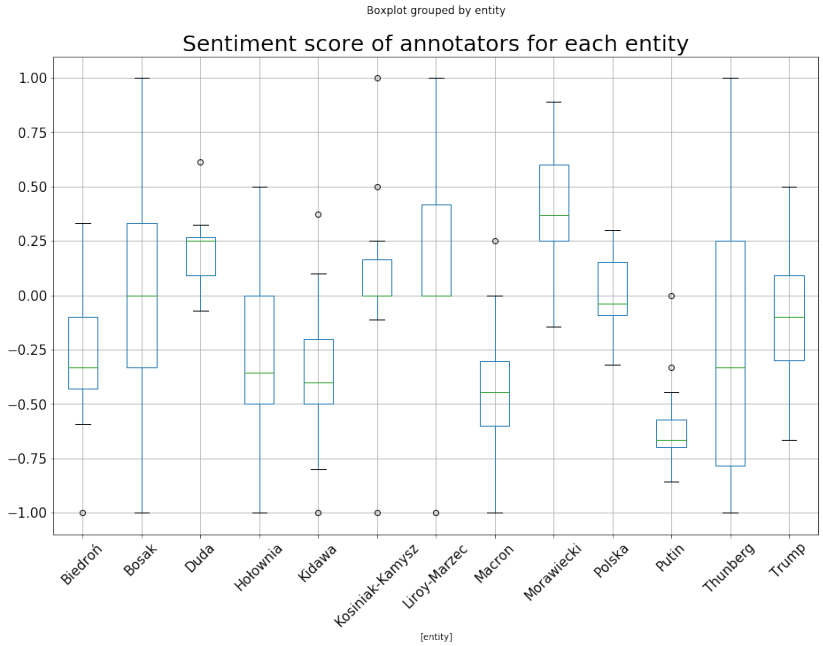

Figure 5.6: Sentiment score of annotators for each entity. We can see that for almost each entity there is an outlier annotator. Entities with less than 10 labels where excluded from this boxplot

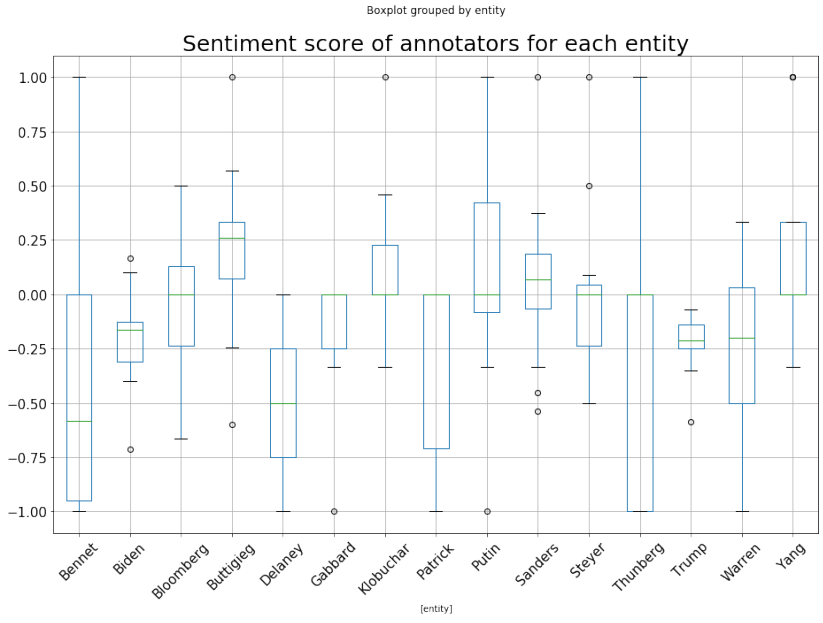

Figure 5.7: Sentiment score of annotators for each entity. We can see that for almost each entity there is an outlier annotator. Entities with less than 10 labels where excluded from this boxplot

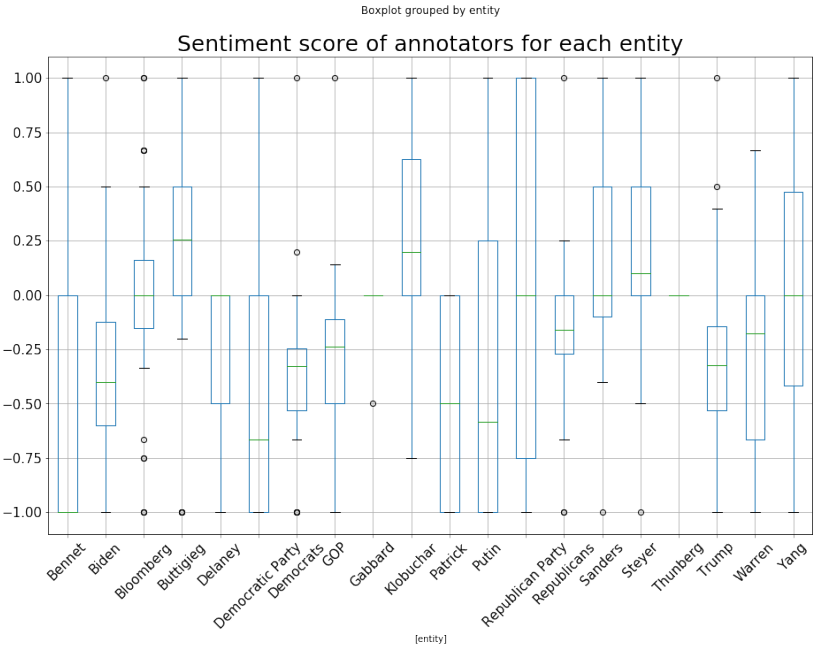

Figure 5.8: Sentiment score of annotators for each entity. We can see that for almost each entity there is an outlier annotator. Entities with less than 10 labels where excluded from this boxplot

entity.

External Datasets: PTB and SEMEVAL-L : Besides the SEN dataset, we run our experiments on two other, publicly available datasets adapted to our problem: PTB – 1042 paragraphs from Portuguese-Brazilian news articles [91] dataset prepared for paragraph-level sentiment analysis consisting of articles about the main president and governor candidates in Brazil and we considered only paragraphs where an entity is mentioned. In addition, since up to our knowledge there are do not exist other dataset about political entities we test models on dataset concerning sentiment about laptops SEMEVAL-L: subset of the semeval14 dataset [92] restricted to 2328 opinions concerning laptops.

5.3.1 Models and techniques used in the experiments

In the next paragraphs we explain the details of techniques used during experiments.

We present a few stages of experiments. In our baseline experiments we use SVM, LSTM and Target-Dependent LSTM. Next we present experiments that use Large Language models. We experiment with BERT, TD BERT and our own model: EntBERT. Then, we present several modifications of BERT: use of all intermediate layers of BERT, gathering embeddings of different tokens and adding external context of the entity. In most cases we address Large Language Model as BERT but in fact, after some minor modifications, any of

Large Language Model can be used there. After each set of models, we present the results and short summary.

5.3.1.1 General experimental setup

All variants of SEN dataset and PTB dataset are randomly split into train (75%) and test (25%) sets. For laptops we use the train test file prepared by the authors of [92]. In all experiments we use 2 evaluation metrics: accuracy and F1 macro-averaged measure. The hyperparameters specific for each model type are provided together with model description. We conducted experiments with several settings and present the best ones.

5.3.1.2 Baseline models

In this place we report baseline experiments with SVM and two LSTM-based models. These models are described in chapter 3. The experimental setup for each model is described in the next paragraphs.

SVM Based on the training set the feature vector space is created using tf-idf algorithm with n-grams for n ranging from 1 to 3. Then the vocabulary from training set, is used to transform test set of headlines to a feature vector. Scikit-learn implementation [93] of svm algorithm is used with the following hyperparameters: kernel is a radial basis function, kernel coefficient gamma is set to 0.5, penalty parameter C of the error term is set to 3, class weight is set to 'balanced' that means that each class has weight calculated as *number of samples / (number of classes * number of class samples))*

LSTM Simple LSTM network is used to classify a sequence of word vectors. Word vectors are created using fasttext[5] algorithm. The whole headline, without modifications is converted into vectors. The following hyperparameters are used: 64 hidden units, weights initialization with uniform distribution from -0.3 to 0.3, learning rate 0.01, clipping gradient at value 200. For word embeddings we pretrained fasttext embeddings on the whole news articles, separately for each language. The network was trained through 50 epochs.

Target-Dependent LSTM (TDLSTM) The model was proposed in [94]. Given text is split into two parts: left and right based on the index of entity. Left part is the sequence of words from the first word to the entity, and right part is the sequence from the entity to the end of the sentence. Both parts are represented as sequences of word embedding vectors. Two LSTM layers are used, one for the left and one for the right part. The last hidden states of the left and right LSTM layers are concatenated. The softmax of the output is calculated to classify the tonality of the text for the given entity as positive, neutral or negative. The hyperparameter settings of the network are analogous as for the LSTM experiments.

[5] `https://fasttext.cc/`

Dataset	metric	svm	LSTM	TDLSTM
SEN-pl	acc	**52.66**	45.09	50.90
	f1-macro	35.66	38.98	**47.70**
SEN-en-R	acc	**54.73**	43.92	45.89
	f1-macro	37.00	34.98	**41.82**
SENenAMT	acc	**44.86**	37.38	39.25
	f1-macro	**43.67**	37.55	39.42
SEN-en	acc	**53.33**	45.12	49.23
	f1-macro	**46.67**	42.49	46.01
PTB	acc	**58.84**	42.30	50.00
	f1-macro	**58.00**	42.25	49.41
SEMEVAL-L	acc	63.13	63.79	**67.08**
	f1-macro	53.00	57.09	**60.90**

Table 5.6: Results for baseline machine learning models. LSTM and TDLSTM models were trained for 50 epochs.

5.3.1.3 The results of the baseline models

First, we present the results of the SVM, LSTM and TDLSTM models in table 5.6. In general, RNNs for news headlines do not improve the results in most cases. SVM is the best among the 3 analysed models according to accuracy metric. It achieves the best accuracy for all datasets with news. The only case where both LSTM based architectures are better is the laptops dataset. When considering F-measure, the TDLSTM model achieves better performance than SVM for SEN-pl and SEN-en-R datasets. For all datasets TDLSTM is better than LSTM. The neural networks are not always better than shallow machine learning. It turns out that for our problem among baselines models, n-grams and svm is better than word embedding and LSTMs.

5.3.1.4 Models based on the transformer architecture

The transformer and BERT architectures are described in chapter 3. In our experiments, we used Hugging Face[6] [95] implementation of BERT. For English "bert base cased"[7] pretrained model was used, for Polish - "polBERT" [8] and for Brazilian Portuguese the model "bert base portuguese cased" [9] described in the paper [96]. Models are trained 10 times for 10 epochs and the best result according to macro f-measure is chosen. The gradient clipping is set to 10. Xavier weight normalisation is used for all dense layers. Batch size is set to 8. Learning rate is 2e-05. Learning rate warmup scheduler is set to 5 steps. L2 normalisation is set to 1e-10. The optimization method is ADAM.

BERT The first layer consists of bidirectional transformer bert model. Embedding of the first token which is the special "$[CLS]$" token is the input for a dense layer ended with softmax.

[6] `https://huggingface.co/`
[7] `https://huggingface.co/bert-base-cased`
[8] `https://huggingface.co/dkleczek/bert-base-polish-uncased-v1`
[9] `https://huggingface.co/neuralmind/bert-base-portuguese-cased`

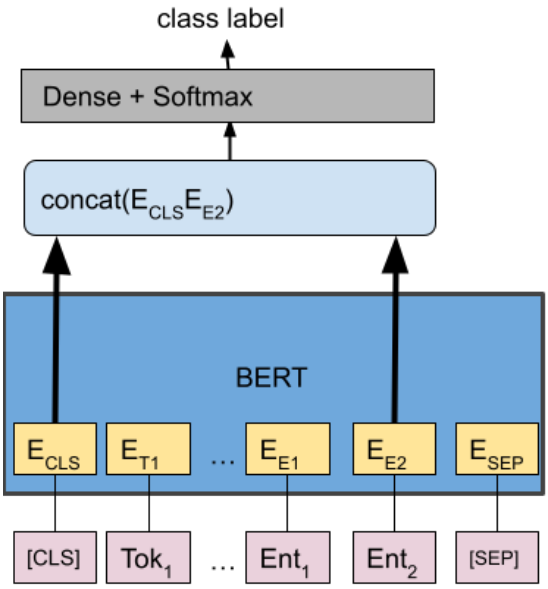

Figure 5.9: EntBERT model visualisation.

Target-Dependent BERT (TDBERT) : is based on work [97] with our custom minor changes. The first part is the same, starting with a BERT layer. Next in the original model the whole BERT embedding of the target part is passed to the pooling layer while we took the embedding of the last target token and passed directly to a linear layer that is followed by softmax function.

EntBERT (our model): We propose a novel modification of BERT called EntBERT (for "Entity BERT") for the entity-level sentiment detection. The first layer is the one with BERT embeddings. Then we concatenate the embedding vector of the first token [CLS] and embedding vector at the entity index of the sentence. Then we use it as the input vector for a linear layer with softmax. The difference to TDBERT is that besides using the token that represents the entity (that is used in TDBERT), we additionally use the token [CLS] that represents the whole sentence. Thanks to that we can use a broader context of the headline. As we show this solution improves the classification performance.

It is also different than the model TD-BERT-QA proposed in th same paper as TDBERT [97] as we do not use helper question to create sentence-pair to create [CLS] token embedding.

The model is presented on figure 5.9

5.3.1.5 The results of BERT-based models

In table 5.7 we present the results of EntBERT, BERT and TDBERT tested on 6 datasets. We can observe that our model EntBERT outperforms other models on the 3 datasets SEN-en-AMT, SEN-en and PTB. It achieves the best

dataset	BERT	EntBERT	TDBERT
SEN-pl acc	61.31±3.18	**61.95±1.57**	61.85±1.89
SEN-pl F1	**56.96±3.2**	55.79±1.62	56.37±2.38
SEN-en-R acc	**54.77±1.75**	52.75±1.83	52.08±1.36
SEN-en-R F1	**44.73±6.35**	43.03±2.59	40.82±1.80
SEN-en-AMT acc	49.04±2.31	**51.86±2.01**	49.22±1.40
SEN-en-AMT F1	47.44±2.36	**50.67±2.02**	48.00±1.26
SEN-en acc	50.79±0.9	**52.85±1.08**	50.97±1.03
SEN-en F1	49.04±1.14	**51.09±1.51**	47.64±1.51
PTB acc	65.61±1.27	**68.07±1.34**	63.11±2.27
PTB F1	65.61±1.65	**67.94±1.31**	62.99±2.23
SEMEVAL-L acc	75.72±0.92	76.34±1.04	**76.51±0.47**
SEMEVAL-L F1	70.16±0.99	71.67±1.84	**72.18±0.89**

Table 5.7: Experimental results for the BERT-based models. Odd rows represent accuracy, even ones represent F1. Each result is the mean after 10 repetitions with random initializations. The best result in each row is typed with the **boldface**.

accuracy for SEN-pl and slighly lower f-measure. On other datasets SEN-en-R and SEMEVAL-L the results of EntBERT are close to the best ones.

5.3.2 The issue of entity sentiment bias

During experiments, we observed that models can unintentionally learn entity bias when predicting the sentiment of the headline i.e. models can learn the sentiment not only based on the context surrounding the entity but also connect it to the entity itself. To solve this problem we used two techniques: masking and substituting the entity with its type. Before applying these techniques, the trained model can exhibit sentiment bias dependent on the given entity (i.e. each headline containing an entity x can be classified as "negative", etc.).

Masking First we replaced the target entity by the special token [MASK].

Substituting entity with its type We tested two approaches with replacing entities with their type. First approach is to replace only the target entity by their type. For example an entity can be replaced by an occupation or function like "politician", "person" or "organisation".

The second approach is as follows. We replace all named entities in the headline with their types.

We used two databses: `DBpedia` https://www.dbpedia.org/ and `spacy`[10]. We found the second database performing better for our experiments, because it has higher entity coverage.

We found replacing entity by type useful not only for removing entity bias tonality, but additionally it enriches the input data with some information about the target.

[10]https://spacy.io/usage/linguistic-features#named-entities

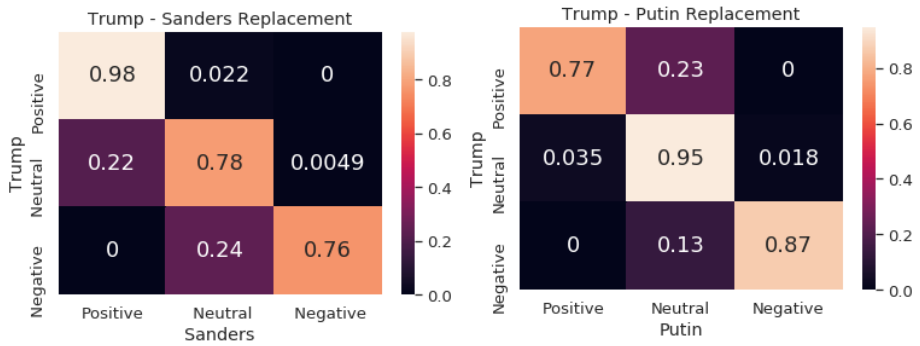

Figure 5.10: Confusion matrices representing the target dependence problem for BERT on the SEN-en dataset. The same headlines with target entity substitution are used for predictions in both confusion matrices. The numbers represent proportions of labels that changed after substituting Trump for Sanders (left) or for Putin (right). Rows represent labels predicted by BERT for headlines where Trump was the target entity, columns the labels for the same headlines where it was substituted. The matrices are row-normalised.

5.3.2.1 Results of entity-bias-aware models

The experimental results for BERT, TDBERT and our EntBERT models, with all 4 variants of entity representations on several datasets are presented in the table 5.8.

Initially we experimented with the unchanged headlines: our proposed EntBERT model for the most datasets is slightly better here than the baselines. However, we discovered an interesting **entity-dependence phenomenon**. To demonstrate this, we made a simple experiment: we substituted the original target entity in the headline with another one (e.g. "Trump" changed to "Sanders", etc.). We observed that the resulting label of the headline was surprisingly often *different* after such substitution (see Table 5.9 for an example). We further investigated this problem , and it turned out that it is omnipresent in case of BERT, EntBERT, and TDBERT models on both SEN-en and SEN-pl datasets.

It seems to introduce a consequent unwanted bias into the classification, i.e. it learns to *"favour"* some entities over other. More detailed figures are presented on Figure 5.10. One can see that the BERT model trained on the SEN-en dataset seems to "favour" the "Sanders" entity over the "Trump" entity, turning many labels from neutral to positive or from negative to neutral with no change of the original headline surrounding the entity.

A possible remedy to this problem is masking the target entity with a special token [MASK] or generalising it with the DBpedia type of the entity. In variant (2) we masked the target entity, in (3) we replace the target entity with its DBpedia type and in (4) we replace all named entities in the headline with their types with the use of `spacy`[11]. One can see, that the variant (2) achieves generally better performance than (1) and (3) even further improves it, while the variant (4) is generally worse than (3).

To summarise: in all the variants (except (4)), our proposed EntBERT model

[11] https://spacy.io/usage/linguistic-features#named-entities

	(1) unmasked entities		
dataset	BERT	EntBERT	TDBERT
SEN-pl acc	61.31±3.18	**61.95±1.57**	61.85±1.89
SEN-pl F1	**56.96±3.2**	55.79±1.62	56.37±2.38
SEN-en-R acc	**54.77±1.75**	52.75±1.83	52.08±1.36
SEN-en-R F1	**44.73±6.35**	43.03±2.59	40.82±1.80
SEN-en-AMT acc	49.04±2.31	**51.86±2.01**	49.22±1.40
SEN-en-AMT F1	47.44±2.36	**50.67±2.02**	48.00±1.26
SEN-en acc	50.79±0.9	**52.85±1.08**	50.97±1.03
SEN-en F1	49.04±1.14	**51.09±1.51**	47.64±1.51
PTB acc	65.61±1.27	**68.07±1.34**	63.11±2.27
PTB F1	65.61±1.65	**67.94±1.31**	62.99±2.23
SEMEVAL-L acc	75.72±0.92	76.34±1.04	**76.51±0.47**
SEMEVAL-L F1	70.16±0.99	71.67±1.84	**72.18±0.89**
	(2) masked target entities		
dataset	BERT	EntBERT	TDBERT
SEN-pl acc	62.51±0.34	**63.04±0.42**	56.43±2.45
SEN-pl F1	56.83±0.31	**58.25±0.86**	48.34±2.46
SEN-en-R acc	**56.39±2.04**	53.80±1.63	51.48±0.50
SEN-en-R F1	**49.87±1.99**	43.21±1.83	48.61±1.11
SEN-en-AMT acc	51.88±1.98	**53.63 ± 1.07**	47.26±1.71
SEN-en-AMT F1	50.71±1.87	**52.84 ± 1.19**	46.13±1.73
SEN-en acc	51.06±1.22	**53.77±0.59**	53.13±1.47
SEN-en F1	49.40±1.40	**51.70±0.61**	48.59±2.50
PTB acc	64.65±3.26	**70.15±1.29**	60.27 ± 2.24
PTB F1	64.47±3.35	**70.05±1.25**	60.27 ± 2.24
SEMEVAL-L acc	75.72±0.90	**76.22± 0.89**	74.81 ± 2.26
SEMEVAL-L F1	70.16±0.98	**71.58 ± 1.22**	70.01± 2.84
	(3) 'typed' target entities		
dataset	BERT	EntBERT	TDBERT
SEN-pl acc	63.60±2.07	**64.61±2.01**	61.01 ± 2.60
SEN-pl F1	57.98±2.43	**59.36±2.09**	56.26 ± 2.95
SEN-en-R acc	**55.85±2.20**	55.13±1.16	52.67 ± 2.99
SEN-en-R F1	**46.56±2.29**	44.61±1.17	42.26 ± 2.08
SEN-en-AMT acc	53.38±2.11	**54.32±2.27**	53.82 ± 1.37
SEN-en-AMT F1	52.44±2.21	**53.33±2.46**	53.14 ± 1.51
SEN-en acc	53.90±1.18	**54.77±1.33**	54.37 ± 0.93
SEN-en F1	52.43±1.09	**52.66±1.37**	52.37±1.48
PTB acc	69.23±1.32	**69.88±2.06**	64.00±1.44
PTB F1	69.17±1.83	**69.74±2.13**	63.74 ± 1.57
	(4) 'typed' all entities		
dataset	BERT	EntBERT	TDBERT
SEN-pl acc	63.23±0.96	**63.30±1.10**	62.10± 1.98
SEN-pl F1	57.70±1.32	**58.46±1.41**	57.40± 2.31
SEN-en-R acc	**57.61±1.77**	56.82±1.38	54.025±1.01
SEN-en-R F1	**50.79±2.34**	47.78±2.05	42.94 ± 1.01
SEN-en-AMT acc	**48.77±1.22**	48.18±0.87	47.04 ± 2.39
SEN-en-AMT F1	**47.57±1.25**	46.96±1.00	45.86 ± 2.32
SEN-en acc	**51.53±0.94**	51.00±1.09	50.08 ± 1.06
SEN-en F1	**49.95±0.87**	48.60±1.33	46.78 ± 1.48

Table 5.8: Experimental results for the 4 variants. Odd rows represent accuracy, even ones represent F1. Each is a mean result after 10 repetitions with random initializations for BERT-based models. The best result in each row is typed with the **boldface**. For each dataset the best score is underlined

.

headline	Entity	Sentiment
entity Arrival in London	Sanders	pos
Brings Controversy but	Putin	neut
Little Surprise	Trump	neg

Table 5.9: An example of the entity-dependence phenomenon. Sentiment label predicted by BERT after the replacement of the target entity with another one changes the label.

is superior to the baseline models on almost all the tested datasets. The highest accuracy and f-measure for 4 datasets SEN-pl, SEN-en-AMT, SEN-en and PTB is achieved by variant (4) and model EntBERT. We were unable to use variant (3) and (4) for SEMEVAL-L dataset because it is not using entities. Also PTB is not included in variant (4).

5.3.2.2 Models based on utilization of all layers

Following the concept of [98] we checked several modifications of BERT utilizing intermediate layers. Hidden layers of BERT store more information than just the last one and we wish to check if adding them to the output improves the overall performance. The two networks LSTM-L-cls and Attention-L-cls are almost the same as in the mentioned publication. The next four are incorporating another modifications to these concepts.

All networks were trained for 10 epochs 10 times with random initialization and the results were averaged. The input contains the masked version of target entity as described in section 5.3.2. The rest of hyperparameters and regularisation is similar as in previous BERT-based models.

LSTM -L-cls This architecture uses an LSTM layer to process hidden BERT's layers representation of [CLS] token: from the first layer to the last one. Then the output of the last hidden state of the LSTM is processed to the dense layer with a softmax function. LSTM with 256 hidden units is used.

Attention-L-cls Instead of LSTM layers we use the attention operation. The dot product attention is used to combine all intermediate layers embeddings at the index of [CLS] token. The result of attention is processed by dense layer with softmax function.

LSTM-L-idx In this model we use LSTM layer that utilize embedding of all BERT's layers but not of the [CLS] token as in LSTM-L-cls but embedding vectors at hte index of entity in a sequence as shown on figure 5.11.

Attention-L-idx This network is very similar to LSTM-L-idx, but instead of an LSTM layer we use self-attention operation to combine all intermediate embedding vector representations of entity. Then the output of the attention is used as input to the dense layer with softmax function.

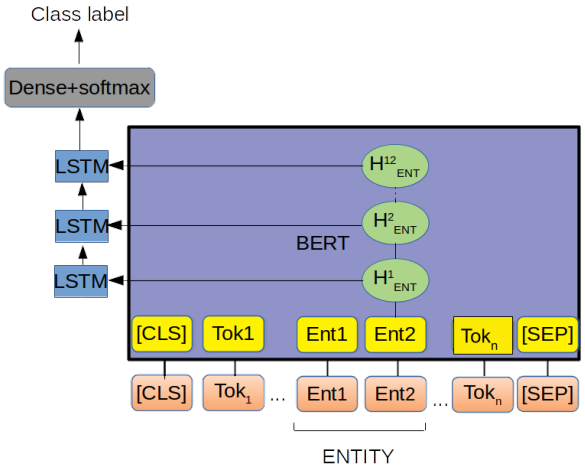

Figure 5.11: LSTM-L-idx, Utilization of embedding from all BERT layers at the index of corresponding to the last entity token

LSTM-L-split In this version we split the input text into two parts by the entity. Then we used them two create two separate BERT embeddings and we again utilize all BERT's layers as input of LSTM layers. This time embedding vector of [CLS] token is used as input because the entity was used for splitting. Then the outputs of the last hidden state of LSTMs are concatenated. The resulting vector is processed through a dense layer with softmax function. The network architecture is presented on figure 5.12.

Attention-L-split This network is very similar to LSTM-L-split, but again the only difference is the attention and dense layer instead of LSTM. We split the input text into two parts by the entity and create two separate BERT embeddings. We utilize all BERT's layers as input of attention. This time embedding vector of [CLS] token is used as the entity was used for splitting. Then the outputs of the attention are concatenated. The resulting vector is processed through a dense layer with softmax function.

5.3.2.3 The results of models based on utilization of all layers

In the table 5.10 we present experimental results of models based on utilization of all BERT layers described in subsection 5.3.2.2. The "type" version of target entity is used. Models that use LSTM to read information from intermediate layers, are better than those using attention except from the Attention-L-split on SEN-pl dataset. Comparing to the previous experiments wich use only the last layer of BERT, the results for SEN-en are better in general but for SEN-pl are worse. The best model for SEN-en is LSTM-L-split with the highest f-measure and slightly lower then the highest accuracy. These results confirm that splitting the input improves the performance despite that the context of the sentence is limited in each part. We have not extended the experiments for

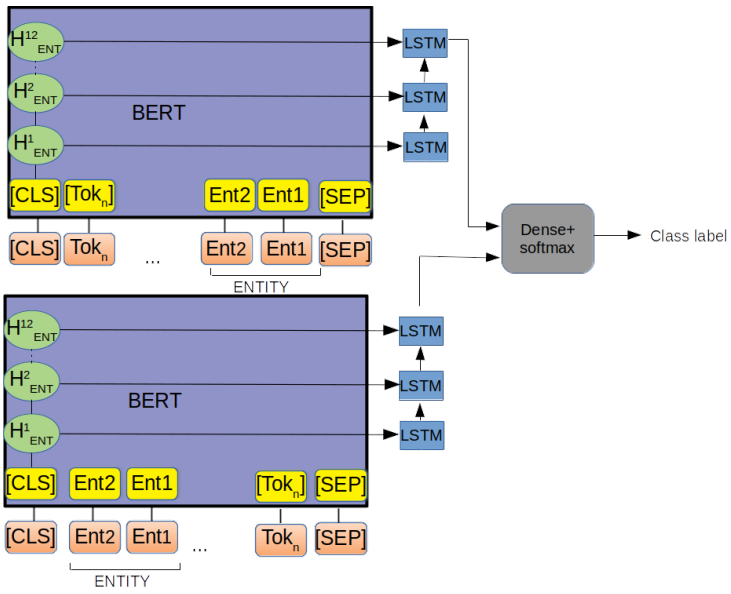

Figure 5.12: LSTM-L-split, Utilization of embedding from all BERT layers at the index of [CLS] token

other datasets as the results were not promising.

5.3.2.4 External context

Pre-trained transformers models are able to embed entities based on the context of the current text and datasets that it was trained on. Additional data can provide useful information for a model, that were not provided in a pretraining step. For example, additional knowledge about target entities, especially for rare entities, can enrich model with information about entities' country, profession, gender or any other important feature. Additional knowledge can be presented directly or using word embeddings.

Wikipedia's assumption is to present neutral information, that means embeddings we used should not be biased by sentiment.

We make a few modifications of networks and inputs to incorporate some external knowledge.

EntSeqBERT BERT transformer CLS token embedding is concatenated with LSTM layer that processes a sequence of entities' embeddings from a given headline. Entities are embedded using Wikipedia2Vec tool [99]. The models for Polish and English language are trained on Wikipedia dump created on June 2022. Wikipedia embedding vector size is 100.

EntSeqBERT2 Again, we used BERT model embedding of CLS token, but we added two LSTM layers that learn about entities. The first LSTM layer learns the sequence of entities in a headline, the second learns the sequence of

dataset	Attention-L-cls	Attention-L-idx	Attention-L-split
SEN-pl acc	56.86 ± 4.30	57.99± 2.41	59.26 ±2.62
f1-macro	51.55 ± 4.56	50.26±3.38	54.52 ± 2.74
SEN-en acc	54.05±2.31	52.15±2.11	52.11 ±3.43
f1-macro	50.00± 2.85	46.63±2.37	49.07 ± 2.74
dataset	LSTM-L-cls	LSTM-L-idx	LSTM-L-split
SEN-pl acc	**59.82 ±1.92**	59.72± 1.33	54.00± 4.44
f1-macro	53.39 ± 3.88	**55.27± 2.67**	43.26 ± 10.83
SEN-en acc	**55.63± 2.30**	55.08±2.03	55.59±1.31
f1-macro	51.85± 2.86	49.70±1.73	**53.27±0.87**

Table 5.10: The results concerning BERT based models utilising all layers averaged over 10 runs with random initialization. The input contains masked target entity version of headline. The best results among all models are **bold**

Dataset	WikiBERT	EntSeqBERT	EntSeqBERT2
SEN-pl acc	**59.65 ± 1.34**	57.41 ± 2.14	56.17 ± 2.8
f1-macro	55.41 ± 1.51	**55.69 ± 2.03**	53.67 ± 4.81
SEN-en-R acc	**53.55 ± 3.06**	50.08 ± 3.15	51.74 ±0.49
f1-macro	**50.84 ± 2.58**	47.43 ± 4.00	50.80 ±0.75
SENenAMT acc	46.42 ± 2.03	47.58 ± 1.60	**51.23 ± 3.05**
f1-macro	44.77 ± 2.29	46.27 ± 1.61	**50.60 ± 2.38**
SENen acc	55.79 ± 3.58	**56.06 ±4.58**	53.68 ± 0.45
f1-macro	52.78 ± 4.23	**55.18 ±0.05**	52.01 ± 0.84

Table 5.11: Mean results after 5 runs with random initialization for BERT models with external context

entities in the whole article. Entities are embedded in the same way as in the previous model.

WikiBERT Two BERT layers are concatenated together with Wikipedia2Vec embeddings. Sentence is split into two parts by entity, which is not included in any part. The first BERT is trained on left part of the sentences, and the second on the right part. Between them there is entity embedding from Wikipedia2Vec.

The learning rate for the models above is set to 1e-05. Gradient clipping is set to 10. The rest of hyperparameters is the same as in previous BERT based models.

5.3.2.5 Results for models with external context

The last presented results 5.11 in this section concern BERT models with external knowledge. Again, the results for SEN-pl are not improved by modifications. For SEN-en the best results among all presented models so far are achieved by EntSeqBERT, the BERT extended by headlines' entities embedding processed by LSTM layer. Among BERT with external knowledge WikiBERT is the best for SEN-pl and SEN-en-R. EntSeqBERT2, BERT extended by wikiembedings of entities from headlines and full articles, is the best one for SENenAMT dataset. In this last experiments entities were typed.

To sum up the presented results , our best models are:

- EntBERT with 'typed' target entities or SEN-pl and SEN-en-AMT datasets,
- WikiBERT for SEN-en-R dataset and EntSeqBERT for SEN-en dataset.
- Also EntBERT gains high results on dataset ptb but this dataset was used only in two first stages of experiments.

In the next section we compare these models with a state-of-the-art model.

5.4 Comparison with state-of-the-art approach

We compared the results on our dataset and on another available dataset newst-mtsc and best models with a model which we called gru-tsc, both presented in the same paper [26]. The results are presented in table 5.12. We used original code provided by the authors to reproduce the results of gru-tsc model on news-mtsc dataset and apply it on SEN dataset.

5.4.1 News-mtsc dataset

This dataset is prepared for multi-target sentiment classification. It was released in the same year as our SEN dataset. It contains sentences from political news articles annotated by MTurk workers. The dataset is split into train with 8739 samples and test set that has two versions called mt and rw. MT consist of 1476 only multi-target sentences and rw, which refers to "real world" and contains 1146 single and multi-target sentences.

5.4.2 Gru-tsc model

In the same paper the authors present a model for target sentiment classification. The model consists of four components that are a pre-trained language model, external knowledge embedding, a target mention mask and bidirectional GRU. There are three model inputs. First input use method suggested in [77] and is a sentence s concatenated with target mentions t and tokenized. This input can be represented as T=$[CLS, s_o, s_1, \ldots, s_p, SEP, t_0, t_1, \ldots, t_q, SEP]$. Second input is a feature representation of a sentence E created using combination of dictionaries as an external knowledge. Third input is a mask M that for each token from input sentence sets 1 if it belongs to the target or 0 otherwise. Each of these three inputs has its own embedding which is next concatenated into one embedding layer. This layer is passed to interaction layer which is build from single BiGRU layer. Then three pooling techniques are used: element wise, mean and maximum is calculated over all hidden states of previous layer. Then these three vectors are stacked and feed into fully connected layer.

5.4.3 Results

The comparison of the gru-tsc model and our best models is presented in 5.12.

We can observe a huge difference in results between SEN and newsmtsc datasets. The results from all models are much higher for news-mtsc data. There may be several reasons for that. First of all, our dataset contains only

dataset	m.	models			
newsmtsc		gru-tsc	WikiBERT	EntBERT	EntSeqBERT
mt	acc	75.25±11.36#*/84.6†	**81.22±0.20**	79.36±0.67	72.95±0.8
mt	f1	70.89±16.40#*/82.5†	**79.71±0.90**	76.77±0.57	71.48±0.9
rw	acc	**82.24±1.37**#/83.8†	82.05±0.54	79.85±0.72	72.07±0.41
rw	f1	**81.45±1.16**#/83.1†	81.37±0.52	79.37±0.75	70.84±0.52
SEN		gru-tsc	WikiBERT	EntBERT	EntSeqBERT
pl	acc	**65.18±12.58** *	59.65±1.34	64.61±2.01	57.41±2.14
pl	f1	51.33±22.00 *	55.41±1.51	**59.36±2.09**	55.699±2.03
en-R	acc	42.77±3.70	53.55±3.06	**55.85±2.20**	50.08±3.15
en-R	f1	29.74±6.04	**50.84±2.58**	46.56±2.29	47.43±4.00
en-AMT	acc	49.16±5.39	46.42±2.03	**54.32±2.27**	47.58±1.60
en-AMT	f1	42.87±8.30	44.77±2.29	**53.33±2.46**	46.27±1.61
en	acc	54.49±2.71	55.79±3.58	54.77±1.33	**56.06±4.58**
en	f1	52.35±2.58	52.78±4.23	52.66±1.37	**55.18±0.05**

Table 5.12: Mean results of gru-tsc model compared with our best models WikiBERT, EntBERT EntSeqBERT on "newsmtsc" and "SEN" data. The notation is as follows: "m." is for metric, two "newsmtsc" datasets "mt" and "rw" denotes "multi-target" and "real world" respectively, for "SEN": datasets "pl" denotes Polish language, "en" English laguage and "R" is for researchers, "AMT" is for Amazon Turk annotators . The best results for each dataset are **bold**. Results marked as †are as reported by the author in the original paper. Results mark as # are as repeated by the author of this thesis based on the original code. Results marked as * are unstable, sometimes the accuracy and f-measure is low.

headlines which may be less clear, more confusing than sentences from articles. The aim of headline is to encourage reading the article rather than provide exact information.

The sentences from newsmtsc were manually selected. Authors removed ambiguous texts. We assumed that in real life news produce headlines and articles of varying quality and ambiguity, so we haven't removed them, only those where no agreement between annotators were achieved.

We compared also the models that have the best performance with proposed gru-tsc model. Our best models are competitive to the state-of-the-art gru-tsc model. For news-mtsc dataset mt version WikiBERT model achieved even better results than results of gru-tsc that we managed to reproduce. For rw version the gru-tsc is just slightly better. The third of our models EntSeqBert achieved the worst performance. For all SEN dataset versions our models achieved better results. Only in SEN-pl the accuracy of gru-tsc is better, but f-measure is much lower than for our models. EntBert achieved the best performance for datasets SEN-pl, SEN-en-R and SEN-en-AMT, EntSeqBert is the best for SEN-en dataset. The gru-tsc has unstable results for SEN-pl and news-mtsc mt, what caused lower results. For news-pl-R dataset f-measure vary from 23.08 to 74.12 and accuracy from 47.65 to 77.65 . For newsmtsc the f-measure is from 37.82 to 80.62 and accuracy from 52.90 to 81.72.

To sum up, our models outperform state-of-the-art model in most of the cases. EntBERT achieved the most promising results but also WikiBERT and EntSeqBert are providing satisfactory results.

5.5 Example application of sentiment detection for news analysis

In this subsection we present possible application of headline tonality classification for media analysis. W used the idea of entity timelines from 4.1 and the model EntBERT trained in 5.3 to present changes of the entity sentiment through time.

To create timelines we use use all crawled headlines that mention given entity from a given period of time. We classify them as positive, negative or neutral relative the given entity using model EntBERT trained on a SEN dataset in a language adequate to a language of news outlet. For English we trained the model on SEN-en and for Polish on SEN-pl.

On a timeline we present the proportion of each headline sentiment class in each month. We can observe how this proportions change through months, from 12.2019 to 06.2020, among different portals. For one entity "Duda" we also manage to crawl older headlines from 11.2014 to 05.2015 from two portals.

Each presented figure stands for one entity and contains separate plots for each news outlet. Each plot contains three lines representing the proportion of a given sentiment class among total number of headlines.

The first two figures 5.13 and 5.14 present the sentiment ratio of "Biden" and "Trump" respectively. The presented period of time is a few months before elections of the USA president on 11.2020. On the first figure "Biden" gains the highest neutral ratio of headlines from wsj.com and nytimes.com ranging from 0.5 to almost 0.8. For nytimes.com the ratio of neutral headlines grows in time and the ratio of negative drops with an increase on the last two months . The most negative headlines about "Biden" are written by truepundit.com with ratio varying from about 0.5 to above 0.6. Also foxnews.com have a high ratio of negative headlines about Biden on the similar level as neutral. Washingtonpost.com produces mixed sentiment of headlines, that is for the most of the time the highest ratio is for neutral but at the beginning af analysed period and at the end there are more negative headlines than positive and in the middle more positive than negative.

On the second figure the sentiment of headlines containing "Trump" is presented. The highest neutral ratio again is achieved by wsj.com and nytimes.com. For truepundit.com we can observe that the ratio of negatives is just slightly higher than neutral for most of the time, in contrast to previous figure where the difference were more noticeable. The most negative headlines are presented by washingtonpost.com, truepundit.com and foxnews.com. All the news outlet keeps the ratio of positive headlines low about 0.1 with some slightly higher ratio for foxnews.com about 0.2.

Two next figures present sentiment ratio for "Democrats" 5.16 and "GOP" 5.15 the two main parties in the USA. In all plots for GOP party we can observe mixed sentiment. Foxnews.com, nytimes.com and truepundit.com present GOP more often in negative light. For wsj.com and washingtonpost.com there is no prevailing sentiment. If two or three points are equal to zero that means we have not found headlines containing provided entity in that month.

The plots are different for "Democrats" entity. We can observe higher proportion of neutral headlines especially for wsj.com, nytimes.com, washingtonpost.com. Two news outlets foxnews.comand truepundit.com have similar pro-

Figure 5.13: Entity-level sentiment timeline for Biden in selected English news outlets.

Figure 5.14: Entity-level sentiment timeline for Trump in selected English news outlets

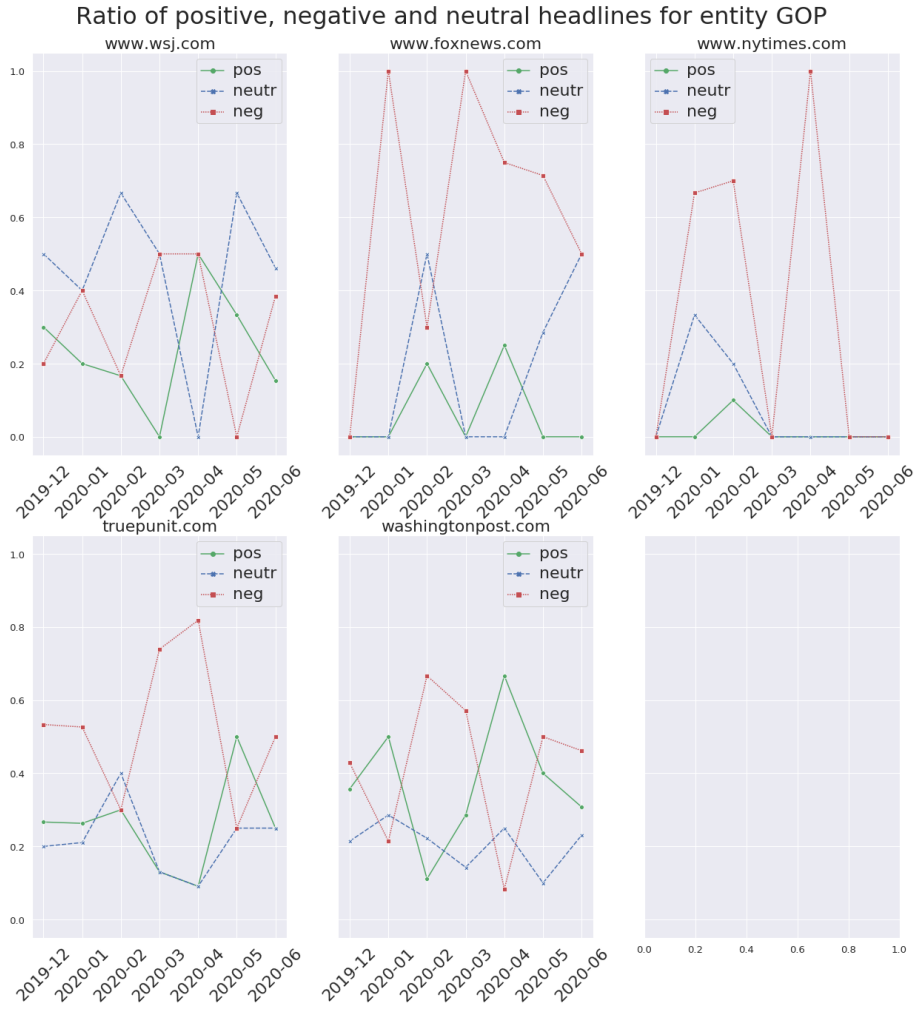

Figure 5.15: Entity-level sentiment timeline for GOP in selected English news outlets

portion of neutral and negative headlines. Truepundit.com have the lowest proportion of positive comments.

To sum up examples of English timelines it seems that some outlets like wsj.com or nytimes.com produce more neutral headlines than others. Another outlet truepundit.com is usually more negative about entities. The proportion of positive headlines is lower for "Trump" than "Biden" except from truepundit.com where both positive ratios are low.

Next two figures present ration of sentiment for entities "Duda" and "Kidawa". This is the time of presidential election in Poland. Entity "Duda" has the highest neutral ratio for all news outlet. The lowest neutral ratio is presented in wpolityce.pl The highest positive ratio of headlines is presented by outlets niezalezna.pl and wplotyce.pl. The rest of the outlets have rather similar ratio of positive and negative headlines containing entity Duda.

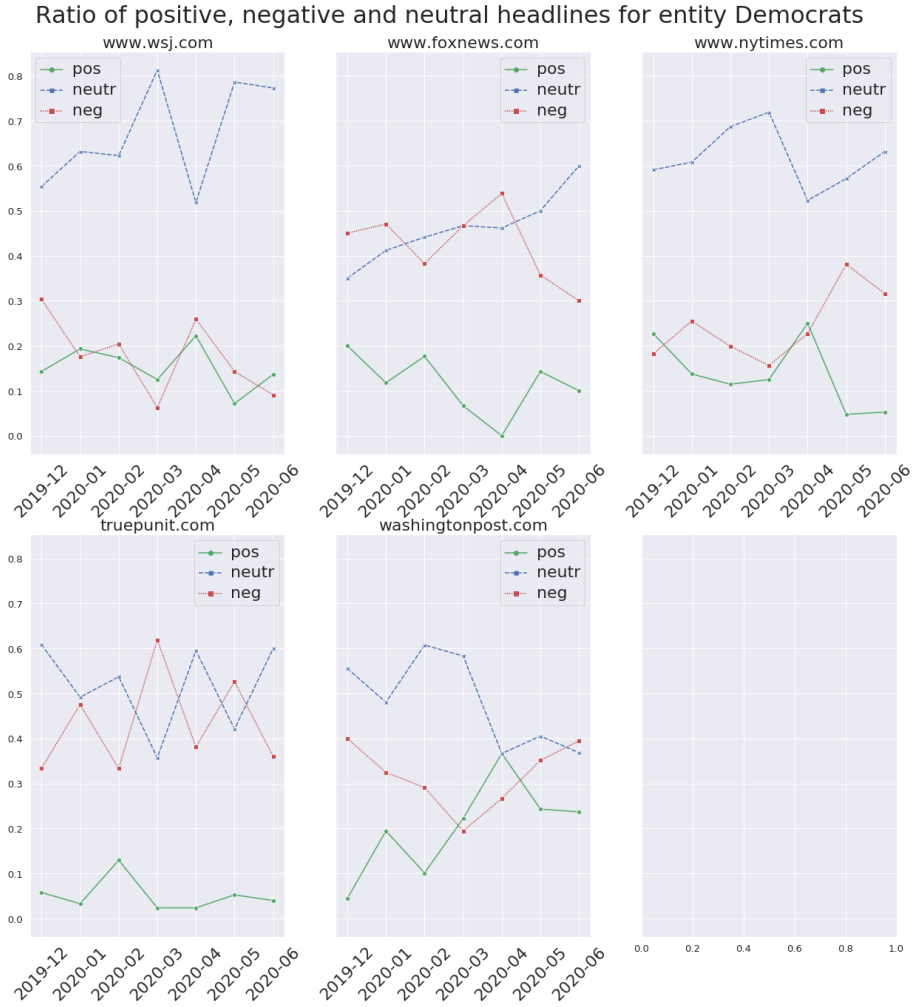

Figure 5.16: Entity-level sentiment timeline for Democrats in selected English news outlets

Figure 5.17: Entity-level sentiment timeline for "Duda" in selected Polish news outlets

Entity "Kidawa" has the highest neutral headlines ratio for almost all of the portals except from wpolityce.pl Also, niezalezna.pl present lower ratio of neutral headlines than most of the outlets. For all the outlets ratio of negative headlines is usually higher than positive headlines.

The last one figure 5.19 present the sentiment ratio for "Duda" from 11.2014 to 05.2015 for two portals. The ratio of neural headlines is the highest for both portals. It can be observed that after 02.2015 the ratio of positive headlines about entity "Duda" increase when negative decreases.

These experiments show exemplary usage of entity-level sentiment analysis. We can observe that even for short period of time there are noticeable, consistent differences between news outlets and the entities they present.

5.6 Summary

In this section we presented the main results of this thesis. In particular we presented the bilingual dataset SEN for entity-level sentiment analysis of news

Figure 5.18: Entity-level sentiment timeline for Kidawa in selected Polish news outlets

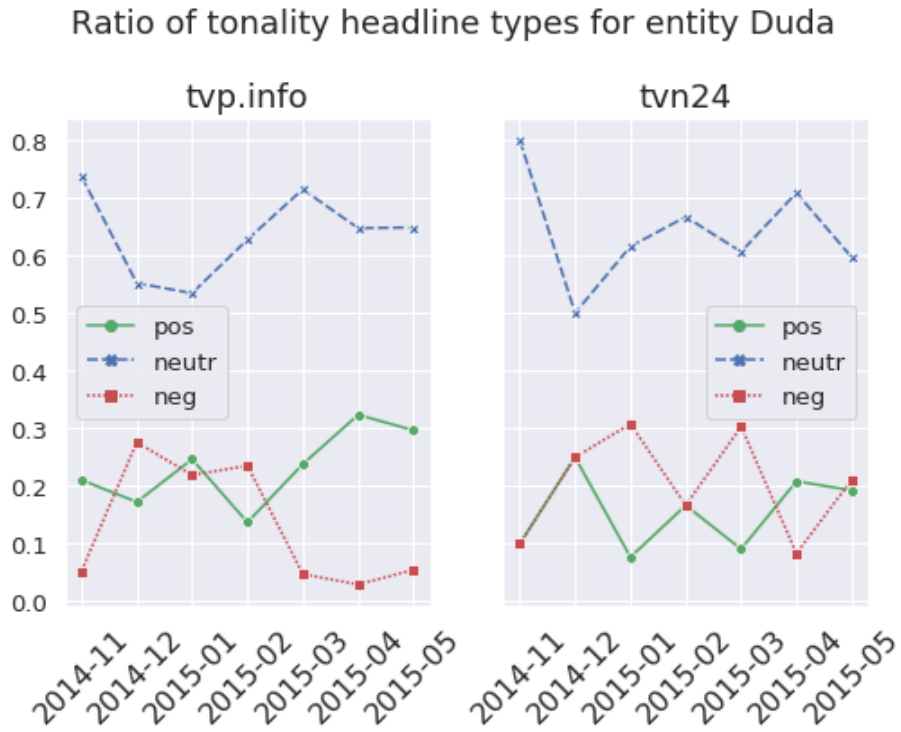

Figure 5.19: Entity-level sentiment timeline for "Duda" in selected Polish news outlets

headlines. According to our knowledge this is the first dataset for Polish language. In addition, we presented extensive experiments on this and other publicly available benchmark datasets. We tested numerous existing and proposed several novel variants of models for entity-level sentiment analysis. The best of our model is EntBERT. Our model outperforms a state-of-the-art model in most of the cases. In addition, we identified entity-bias phenomenon and proposed some techniques for removing it that additionally improved the performance of the models.

Chapter 6

Persuasion techniques

In this chapter we study another problem that is related to media bias analysis, namely the problem of persuasion techniques detection. We presented a novel variant of model for this task and experimental results on a publicly available dataset. The results presented in this chapter were submitted to the Semeval 23 international competition [100] (task 3).

6.1 Motivation

News articles are expected to present reliable information, however quite often contain some kind of manipulation. Unconscious reader can be unable to spot all kind of persuasion that he is exposed to. Reading such news can influence a reader's point of view especially if the reader has a low level of political knowledge [7]. Automatic tools for detection of persuasion can help in media analysis and creating more objective news.

We used deep learning model for persuasion techniques detection in news articles.

6.2 Problem description

We focus on paragraph-level persuasion techniques recognition. We created model trained and tested on languages: English, French, German, Italian, Polish and Russian and for additional languages with only test data: Spanish, Greek and Georgian.

Our solution combines multitask learning and hierarchical neural networks. The system identifies span where the persuasion techniques appear and then passes this information to the next module that classifies it to one or more categories of persuasion techniques. We used pre-trained and fine-tuned Bert embedding as the shared input layer.

We discover that adding an auxiliary task such as span identification as the first step of training neural network may give better results than simply classifying the whole paragraph. Moreover, using one neural network for that approach makes solving the task possible even for limited data.

The code of our solution is publicly available at `https://github.com/Katarzynaa/persuasion_detection`

Technique	N. test	N. train
Doubt	187	518
Whataboutism	2	16
Appeal to Hypocr.	8	40
Causal Oversimp.	24	213
Appeal to Author.	28	154
Guilt by Associat.	4	59
Slogans	28	153
Flag Waving	96	287
Loaded Lang.	483	1809
Red Herring	19	44
False Dil.-NoCh.	63	122
App. to Popular.	34	15
Convers. Killer	25	91
Name Call.-Lab.	250	979
A.to Fear-Prejud.	137	310
Exaggerat.-Mini.	115	466
Repetition	141	544
Straw Man	9	15
Obf.-Vag.-Conf.	13	8

Table 6.1: Number of labels in English dataset

The problem is the multi-label classification task, i.e. to classify all persuasion techniques that appear in every paragraph of a news article. The input data is a single paragraph, the output is a list of detected techniques. The data is provided in three folders: "train", "dev" and "test". Each folder contains files with articles. Each article is divided into paragraphs separated by an additional empty line. For each paragraph of each article in "train" and "dev" sets organisers provide the lists of labels of persuasion techniques in separate files. Labels of test data are unknown. As an additional information authors provide for each paragraph the list of spans, start and end character index, and its class.

The data can be downloaded from `https://propaganda.math.unipd.it/semeval2023task3/`. There are 23 classes with a distribution that is not balanced (far from the uniform one). These classes are presented in table 6.1 One can find a detailed description in the task description paper [100].

6.3 Multitask Hierarchical Networks for persuasion techniques identification

Our solution is based on multitask networks. This kind of network shares the same part or full architecture to solve several tasks being trained at once. Much research evidence demonstrates that properly choosing of auxiliary task may help to get better results at the main one (e.g. [101]).

Multi-task learning is mostly used for emotion and sentiment analysis [102], aspect based sentiment analysis [103] or named entity recognition, etc.

Hierarchical networks are formed as an acyclic graph, which means that the tasks are learned by the networks' modules in some order. The results of

previous modules influence the next modules. Hierarchical multitask approach has many forms and applications in NLP, for example: embedding learning [104] or aspect-based sentiment analysis [105], etc. In our solution we created a network that solves two tasks: span identification and persuasion techniques identification. Both share the same input of Bert embedding but have separate layers for the classification part. The second one uses the results of the first one. Span identification can be treated as a sequence tag classification. We predict the span where the persuasion technique is present. For the second task we used the predicted span to identify the technique used in that span. If more than one span is identified we predicted techniques based on the first index. One span can represent many persuasion techniques.

We also experimented with other modifications or variants of network architectures. For example, we tried to add auxiliary task of POS tagging, adding general sentiment based on "Vader" approach [106], add entity-level sentiment trained on the SEN benchmark dataset [5] however no improvement was observed so details are not included.

In our preliminary approach we tried to translate articles between languages to get more data but no improvement was observed. That can be caused by the fact that specific persuasion techniques for a given language are not easy to translate to other language or the fact that a basic translator omits persuasion techniques and translates it in more neutral words.

6.4 System overview

Our system proceeds in the following general steps:

1. Use pre-trained model to continue pre-training using masked language model task on provided news articles data from all subtasks
2. Train the multitask hierarchical model on specific tasks: span detection and multilabel classification
3. Evaluate model on the devset
4. Train model on joint trainset and devset

For the step 1 we used standard code provided by huggingface[1]. Step 2 uses our implementation of a multitask hierarchical BERT based network that is described in the next subsection.

One approach is used for all languages that contain train and dev set but models are trained separately.

For languages that do not have train and dev set we translate the articles from Polish to the required language. For these languages we use simple BERT model for sequence classification. We do not use our model as we believe it is hard to find exact spans after translation.

6.4.1 Model architecture

General schema of the model architecture is presented in Figure 6.1.

The first layer of a model is a BERT layer. It takes tokenized input paragraph and calculates embeddings. Next we add dropout layer. Then it is followed by the first linear layer responsible for tag classification for span identification. We

[1] `https://huggingface.co/`

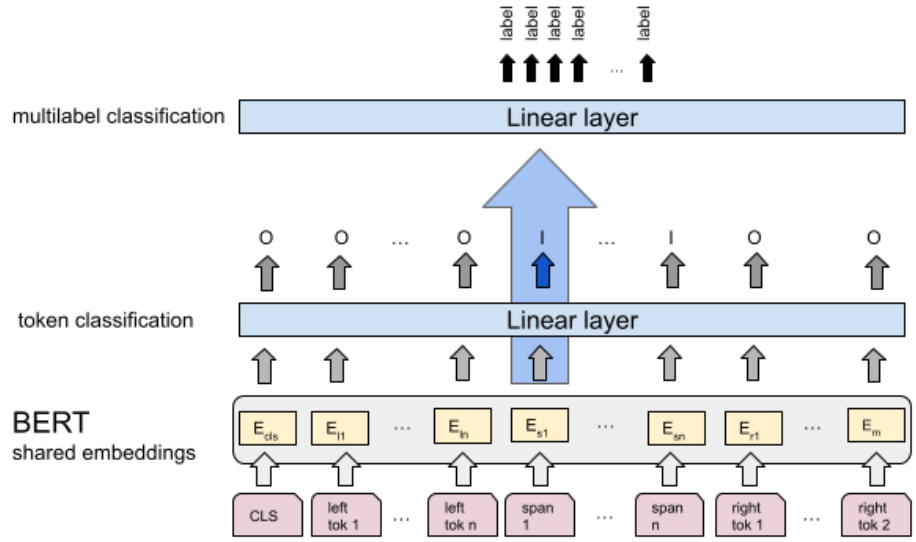

Figure 6.1: Proposed model architecture. Bert embedding layer is shared between two linear layers. First linear layer identifies the first token index (dark blue arrow) of persuasion text span. Then its embedding of the first token of persuasion span is passed to the second linear layer that performs multilabel classification. The index of span embedding even for the same sample may change during training.

use softmax as activation function of this layer. It classifies each token as *I* or *O*.

We take the index of the first token of a span and we use it to find the BERT embedding from this token from the previous layer. Then this embedding is passed to the second linear layer. If no span is identified the layer takes the first token of BERT embedding. The second linear layer is followed by sigmoidal activation function. It has as many outputs as the number of classes and determines whether the sample belongs to each class or not (it can activate for any class).

Loss function for our model is a sum of two:

$$Loss = 0.5 * Loss_1 + Loss_2 \tag{6.1}$$

The $Loss_1$ is cross entropy that calculates the loss for span identification and is used without any modification.

$$Loss1_n = -[y_n log x_n + (1 - y_n) * log(1 - x_n)] \tag{6.2}$$

where y_n is the correct answer and x_n is the prediction for the nth batch. The $Loss_2$ is used for a multi-label classification and is binary cross entropy calculated for each label with added weight p_c:

$$Loss2_{n,c} = -[p_c y_n log x_n + (1 - y_n) * log(1 - x_n)] \tag{6.3}$$

We add weight p_c because of unbalanced number of classes. Weight of a

class c is a proportion of samples from other classes to the number of samples from class c calculated as follows:

$$p_c = \frac{n_{all} - n_c}{n_c} \tag{6.4}$$

where n_{all} is the number of all samples in the dataset and n_c of the current class.

6.5 Experimental Setup

For final evaluation we use train file for training and test file for evaluation. After choosing the right parameters we train the whole model on both datasets.

We used random search for hyper-parameter tuning. For final model we used hyper-parameters as follows. Maximum length of a sequence is 256, as it speeds up the computation and we did not see much improvement when using longer sequences. Train batch size is 8 as it is mostly limited by the machines that were available. Maximum number of epochs is 30 and we saved the best model on dev set according to the micro-f1 metric. Learning rate was set to 1e-05 and max gradient norm was set to 10. Dropout before both linear layers is set to 0.1. The network is trained using AdamW optimization algorithm with weight decay equal to 0.01.

6.5.1 Data preprocessing

Spans are provided as a list of starting and ending indexes of characters. We converted them to list of I and O tags where each tag corresponds to one word. Tag O means that the word does not contain persuasion and I means that word belongs to a persuasion span. First we identify which parts are between start and end index and then we split into words using the "spacy" library[2].

When preparing the input data to the BERT model, transformers have their own tokenizer and it may split the words into smaller tokens. In such case each token has the same tag as the original word.

All labels were provided as a list. We convert them to multi-hot encoding vectors. Pre-trained models from table 6.2 are used in our retraining . All of them come from the huggingface[3]. We use all sets from all the three tasks to continue pre-training our BERT model for a particular language. We use masked language model for that.

Models were evaluated by $micro - F_1$ in a first place and then $macro - F_1$.

6.6 Results

The results on the official test set are presented in table 6.4. Our system for all languages performs better than the baseline. Interestingly, models that have no train and dev set such as Spanish, Greek or Georgian still perform better than baseline but much worse than other languages. That means we can achieve better performance using translation but it is never good enough. The reason

[2] `https://spacy.io/`
[3] `https://huggingface.co/`

Language	model name
English	bert-base-cased
Polish	dkleczek/bert-base-polish-uncased-v1
French	dbmdz/bert-base-french-europeana-cased
Italian	dbmdz/bert-base-italian-uncased
Russian	DeepPavlov/bert-base-bg-cs-pl-ru-cased
German	dbmdz/bert-base-german-uncased
Spanish	dccuchile/bert-base-spanish-wwm-uncased
Greek	nlpaueb/bert-base-greek-uncased-v1
Georgian	bert-base-multilingual-cased

Table 6.2: Transformer models used for continuing pre-training.

		Our model		baseline	
Lang	rank/total	f-micro	f-macro	f-micro	f-macro
Polish	11/20	0.31427	0.17940	0.17928	0.05932
French	11/20	0.36246	0.26628	0.24014	0.09867
Italian	15/20	0.39874	0.20056	0.39719	0.12152
Russian	12/19	0.25289	0.11653	0.20722	0.08598
German	12/20	0.37264	0.20109	0.31667	0.08345
English	23/23 12*	0.06022 0.30113*	0.03066 0.08047*	0.19517	0.06925
Spanish	13/17	0.24490	0.14257	0.24843	0.02007
Greek	12/16	0.15021	0.12095	0.08831	0.00606
Georgian	13/16	0.15017	0.09988	0.13793	0.14083

Table 6.3: Results on the final test sets. For English language we present post-evaluation result marked as $*$, as the reason of the lower score on official evaluation is a wrong file uploaded. The rank with * is calculated based on official leader board. rank/total number of participants, baseline - official baseline, svm model with unigrams and bigrams as input

could be that translation tries to preserve the meaning but may skip or change the type of persuasion. Also the techniques for different languages may vary.

All models on the dev set perform much better than the baseline (Figure 6.3). According to micro-f1, the results for 3 random runs of model are stable, the standard deviation is rather low. According to macro-f1, the standard deviation is slightly higher, that means one can observe differences between dev set results and test set results. The reason may be that the data for test set may come from different topics and time. The system learns the techniques that are more specific for train and dev set.

6.6.1 Error analysis

We analysed which classes in the English devset are the easiest/hardest to be recognized (Table 6.5). The most frequent class ("loaded language") gets a high f-measure value, but the highest one is achieved by the “Guilt by Association” class which has only 4 and 59 observations in the test and train sets, respectively, which means it is probably the easiest class to detect. Classes "Loaded Lan-

	Our model		BERT		baseline	
Lang.	f-micro	f-macro	f-micro	f-macro	f-micro	f-macro
English	40.80±0.1	16.04±3.0	38.11±1.2	14.54±0.3	16.13	21.74
Polish	36.72±0.6	21.13±2.3	36.24±0.2	21.81±0.8	12.52	5.67
French	41.57±0.9	28.25±1.4	38.59±0.5	28.49±0.6	29.29	13.48
Italian	43.51±2.0	22.07±0.5	40.22±2.8	20.51±0.6	38.92	10.39
Russian	44.60±0.4	15.98±2.0	39.97±2.0	14.69±2.0	25.32	4.284
German	41.20±0.1	23.73±1.6	39.64±0.1	23.59±1.0	33.12	10.02

Table 6.4: Mean scores achieved on the dev set for all language. Our model for each language was run 3 times. For baseline we get the results from the leader board

guage" and "Name-Calling Labeling" get high recall but lower precision, which means they are often wrongly detected. "False Dillema-No Choice" gets much higher precision than recall which means it is precisely recognized. There are a few classes that are not recognized at all and most of them have the low number of samples.

Some classes may require another approach, like adding broader context. For example "Red Herring" is when someone introduces irrelevant information, what may be hard to detect based on a single sentence like *He died there.* what was classified wrongly as "Repetition" or *"Melania paired the mid-length half price frock with Christian Loubotin heels"* what was classified as "Loaded language" and "Name-Calling-Labelling". Both cases are hard to be recognized without the context. For example, the second is suited to an article about fashion but not about politics.

The system was wrong also about the "Conversation Killer" technique which is often a short and rather obvious statement: *"Everybody knows it."* or hidden in some long paragraph *"How about sorting that stuff out instead of politicizing something that should be fun for everyone? How many times does it have to be said".*

We noticed that sometimes a broader context not only from the article but also from the world of politics is necessary to correctly recognise a technique. For example, the paragraph containing "Appeal to Hypocrisy" *But he didn't mention Mueller for the rest of the day.* and *Of course, Sir Kim would have had plenty of targets had he decided to pass judgement on the present incumbent of the White House.* are not easy to be classified based only on what is given in the sentence.

Only 129 paragraphs were predicted correctly from the English devset (all true labels are correctly recognised and no other labels). Most of them have one or two labels.

We discovered that simple change of the index in Bert embedding may help to improve the persuasion classification. Moreover, we are able to identify spans and perform classification on limited data using the described networks. Our system works better than classic BERT for sequence classification and can be used as a technique for persuasion detection in news.

Technique	precision	recall	f1
Doubt	0.29	0.25	0.27
Whataboutism	0.00	0.00	0.00
Appeal to Hypocr.	0.00	0.00	0.00
Causal Oversimp.	0.06	0.08	0.07
Appeal to Author.	0.10	0.04	0.05
Guilt by Associat.	0.60	0.75	0.67
Slogans	0.26	0.25	0.25
Flag Waving	0.46	0.50	0.48
Loaded Lang.	0.49	0.89	0.63
Red Herring	0.00	0.00	0.00
False Dil.-NoCh.	0.30	0.05	0.08
App. to Popular.	0.00	0.00	0.00
Convers. Killer	0.00	0.00	0.00
Name Call.-Lab.	0.39	0.70	0.50
A.to Fear-Prejud.	0.26	0.15	0.19
Exaggerat.-Mini.	0.19	0.38	0.26
Repetition	0.19	0.04	0.06
Straw Man	0.00	0.00	0.00
Obf.-Vag.-Conf.	0.00	0.00	0.00

Table 6.5: Scores for each class achieved by our model on English devset . Prec, rec, f1 are precision recall and f1-measure respectively.

6.7 Summary

In this chapter we presented additional study concerning the problem of persuasion detection techniques. The hierarchical multitask neural network was proposed. The experiments were conducted on several languages. The results show that our solution is better than the baseline. These experiments may help to detect that the text of the article contains phrases that manipulate the reader which is another way to reveal bias.

Part III

Summary and conclusions

At the end of this thesis we would like to summarise the main contributions and achievements in the order of their importance.

In chapter 5 we proposed transformer-based models for entity-level sentiment analysis. We reproduced the state-of-the-art models. Our models achieved higher f-measure than state-of-the-art gru-tsc model for all SEN dataset variants and partially for newsmtsc dataset. Only for newsmtsc "mt" variant the gru-tsc model was better.

We discovered that the models, learning to detect sentiment, become biased towards some entities. Several solutions to the problem were proposed: masking the target entity, replacing the target entity by its type and replacing all occurrences of the entities in headline by their type. These methods not only overcame the problem of entity-bias but also improved the models' performance.

In chapter 6 we proposed a hierarchical multitask neural network architecture for persuasion techniques detection. The network performed two tasks: it identified the span of the persuasion and then detected the technique. The model was able to learn the main task together with auxiliary task, what is faster and more efficient than training two models separately, especially when the data is limited. Moreover, shared weights helped to learn the main task faster and more accurately.

In chapter 4 we also presented some helper techniques like finding articles describing similar topics and entity timeline analysis. The timeline together with sentiment analysis is presented as a prototype of a tool for media analysis.

We presented methods that can help other researchers and society to analyse media bias. We believe that our experiments will be inspiration for others to improve the quality of news presented in digital media.

List of Figures

List of Tables

Bibliography

[1] Raymond S Nickerson. Confirmation bias: A ubiquitous phenomenon in many guises. *Review of general psychology*, 2(2):175–220, 1998.

[2] Eli Pariser. *The filter bubble: What the Internet is hiding from you*. Penguin UK, 2011.

[3] Katarzyna Baraniak and Marcin Sydow. *Towards Entity Timeline Analysis in Polish Political News*, pages 323–332. Springer International Publishing, Cham, 2019.

[4] Katarzyna Baraniak and Marcin Sydow. News articles similarity for automatic media bias detection in polish news portals. In M. Ganzha, L. Maciaszek, and M. Paprzycki, editors, *Proceedings of the 2018 Federated Conference on Computer Science and Information Systems*, volume 15 of *Annals of Computer Science and Information Systems*, pages 21–24. IEEE, 2018.

[5] Katarzyna Baraniak and Marcin Sydow. A dataset for sentiment analysis of entities in news headlines (sen). *Procedia Computer Science*, 192:3627–3636, 2021. Knowledge-Based and Intelligent Information and Engineering Systems: Proceedings of the 25th International Conference KES2021.

[6] Katarzyna Baraniak and Marcin Sydow. Kb at semeval-2023 task 3: On multitask hierarchical bert base neural network for multi-label persuasion techniques detection. In *Proceedings of the 17th International Workshop on Semantic Evaluation*, SemEval 2023, Toronto, Canada, July 2023.

[7] Jakob-Moritz Eberl, Hajo G Boomgaarden, and Markus Wagner. One bias fits all? three types of media bias and their effects on party preferences. *Communication Research*, 44(8):1125–1148, 2017.

[8] Karthik Sheshadri, Chung-Wei Hang, and Munindar Singh. The causal link between news framing and legislation. *arXiv preprint arXiv:1802.05768*, 2018.

[9] Marta Recasens, Cristian Danescu-Niculescu-Mizil, and Dan Jurafsky. Linguistic models for analyzing and detecting biased language. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1650–1659, 2013.

[10] Jakub Piskorski, Marcin Sydow, and Dawid Weiss. Exploring linguistic features for web spam detection: a preliminary study. In *AIRWeb '08:*

Proceedings of the 4th international workshop on Adversarial information retrieval on the web, pages 25–28, New York, NY, USA, 2008. ACM.

[11] Diego Gomez-Zara, Miriam Boon, and Larry Birnbaum. Who is the hero, the villain, and the victim?: Detection of roles in news articles using natural language techniques. In *23rd International Conference on Intelligent User Interfaces*, pages 311–315. ACM, 2018.

[12] Konstantina Lazaridou and Ralf Krestel. Identifying political bias in news articles. *Bulletin of the IEEE TCDL*, 12, 2016.

[13] Haokai Lu, James Caverlee, and Wei Niu. Biaswatch: A lightweight system for discovering and tracking topic-sensitive opinion bias in social media. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 213–222. ACM, 2015.

[14] Johannes Kiesel, Maria Mestre, Rishabh Shukla, Emmanuel Vincent, Payam Adineh, David Corney, Benno Stein, and Martin Potthast. SemEval-2019 Task 4: Hyperpartisan News Detection. In *12th International Workshop on Semantic Evaluation (SemEval 2019)*. Association for Computational Linguistics, June 2019.

[15] Ceren Budak, Sharad Goel, and Justin M Rao. Fair and balanced? quantifying media bias through crowdsourced content analysis. *Public Opinion Quarterly*, 80(S1):250–271, 2016.

[16] Felix Hamborg. *Media Bias Analysis*, pages 11–53. Springer Nature Switzerland, Cham, 2023.

[17] Tim Althoff, Xin Luna Dong, Kevin Murphy, Safa Alai, Van Dang, and Wei Zhang. Timemachine: Timeline generation for knowledge-base entities. *CoRR*, abs/1502.04662, 2015.

[18] Arturas Mazeika, Tomasz Tylenda, and Gerhard Weikum. Entity timelines: visual analytics and named entity evolution. In Craig Macdonald, Iadh Ounis, and Ian Ruthven, editors, *CIKM*, pages 2585–2588. ACM, 2011.

[19] Pedro Saleiro, Jorge Teixeira, Carlos Soares, and Eugénio Oliveira. Timemachine: Entity-centric search and visualization of news archives. In *European Conference on Information Retrieval*, pages 845–848. Springer, 2016.

[20] Dafna Shahaf and Carlos Guestrin. Connecting the dots between news articles. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '10, pages 623–632, New York, NY, USA, 2010. ACM.

[21] Paul Neculoiu, Maarten Versteegh, and Mihai Rotaru. Learning text similarity with siamese recurrent networks. In *Proceedings of the 1st Workshop on Representation Learning for NLP*, pages 148–157, 2016.

[22] Anna Huang. Similarity measures for text document clustering. In *Proceedings of the sixth new zealand computer science research student conference (NZCSRSC2008), Christchurch, New Zealand*, pages 49–56, 2008.

[23] Nava Tintarev and Judith Masthoff. Similarity for news recommender systems. In *Proceedings of the AH'06 Workshop on Recommender Systems and Intelligent User Interfaces*. Citeseer, 2006.

[24] Kathleen R McKeown, Regina Barzilay, David Evans, Vasileios Hatzivassiloglou, Judith L Klavans, Ani Nenkova, Carl Sable, Barry Schiffman, and Sergey Sigelman. Tracking and summarizing news on a daily basis with columbia's newsblaster. In *Proceedings of the second international conference on Human Language Technology Research*, pages 280–285. Morgan Kaufmann Publishers Inc., 2002.

[25] Felix Hamborg, Karsten Donnay, and Bela Gipp. Towards target-dependent sentiment classification in news articles. In *Proceedings of the iConference 2021*, Mar. 2021.

[26] Felix Hamborg and Karsten Donnay. NewsMTSC: A dataset for (multi-)target-dependent sentiment classification in political news articles. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 1663–1675, Online, April 2021. Association for Computational Linguistics.

[27] Ralf Steinberger, Stefanie Hegele, Hristo Tanev, and Leonida Della Rocca. Large-scale news entity sentiment analysis. In *Proceedings of the International Conference Recent Advances in Natural Language Processing, RANLP 2017*, pages 707–715, Varna, Bulgaria, September 2017. INCOMA Ltd.

[28] Gabriel Domingos de Arruda, Norton Trevisan Roman, and Ana Maria Monteiro. An annotated corpus for sentiment analysis in political news. In *Proceedings of the 10th Brazilian Symposium in Information and Human Language Technology*, pages 101–110, Natal, Brazil, November 2015. Sociedade Brasileira de Computação.

[29] Julio Cesar Soares Dos Rieis, Fabrício Benevenuto de Souza, Pedro Olmo S Vaz de Melo, Raquel Oliveira Prates, Haewoon Kwak, and Jisun An. Breaking the news: First impressions matter on online news. In *Ninth International AAAI conference on web and social media*, 2015.

[30] Gregory Grefenstette, Yan Qu, James G. Shanahan, and David A. Evans. Coupling niche browsers and affect analysis for an opinion mining application. In *Coupling Approaches, Coupling Media and Coupling Languages for Information Retrieval*, RIAO '04, page 186–194, Paris, FRA, 2004. LE CENTRE DE HAUTES ETUDES INTERNATIONALES D'INFORMATIQUE DOCUMENTAIRE.

[31] Alexandra Balahur and Ralf Steinberger. Rethinking sentiment analysis in the news: from theory to practice and back. *Proceeding of WOMSA*, 9, 2009.

[32] Alexandra Balahur, Ralf Steinberger, Mijail Kabadjov, Vanni Zavarella, Erik van der Goot, Matina Halkia, Bruno Pouliquen, and Jenya Belyaeva. Sentiment analysis in the news. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, Valletta, Malta, May 2010. European Language Resources Association (ELRA).

[33] Lisa Fan, Marshall White, Eva Sharma, Ruisi Su, Prafulla Kumar Choubey, Ruihong Huang, and Lu Wang. In plain sight: Media bias through the lens of factual reporting. *arXiv preprint arXiv:1909.02670*, 2019.

[34] Saif Mohammad, Svetlana Kiritchenko, Parinaz Sobhani, Xiaodan Zhu, and Colin Cherry. Semeval-2016 task 6: Detecting stance in tweets. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 31–41, 2016.

[35] Wei-Fan Chen, Henning Wachsmuth, Khalid Al Khatib, and Benno Stein. Learning to flip the bias of news headlines. In *Proceedings of the 11th International Conference on Natural Language Generation*, pages 79–88, 2018.

[36] Reid Pryzant, Richard Diehl Martinez, Nathan Dass, Sadao Kurohashi, Dan Jurafsky, and Diyi Yang. Automatically neutralizing subjective bias in text. *arXiv preprint arXiv:1911.09709*, 2019.

[37] Namrata Godbole, Manja Srinivasaiah, and Steven Skiena. Large-scale sentiment analysis for news and blogs. *Icwsm*, 7(21):219–222, 2007.

[38] Aleksander Wawer and Julita Sobiczewska. Predicting sentiment of polish language short texts. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2019)*, pages 1321–1327, 2019.

[39] Aleksander Wawer. Sentiment analysis for polish. *Poznan Studies in Contemporary Linguistics*, 55(2):445–468, 2019.

[40] Jan Kocoń, Piotr Miłkowski, and Monika Zaśko-Zielińska. Multi-level sentiment analysis of polemo 2.0: Extended corpus of multi-domain consumer reviews. In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, pages 980–991, 2019.

[41] Dayan de França Costa and Nadia Felix Felipe da Silva. Inf-ufg at fiqa 2018 task 1: Predicting sentiments and aspects on financial tweets and news headlines. In *Companion Proceedings of the The Web Conference 2018*, WWW '18, page 1967–1971, Republic and Canton of Geneva, CHE, 2018. International World Wide Web Conferences Steering Committee.

[42] Simra Shahid, Shivangi Singhal, Debanjan Mahata, Ponnurangam Kumaraguru, Rajiv Ratn Shah, et al. Aspect-based sentiment analysis of financial headlines and microblogs. In *Deep Learning-Based Approaches for Sentiment Analysis*, pages 111–137. Springer, 2020.

[43] Keith Cortis, André Freitas, Tobias Daudert, Manuela Huerlimann, Manel Zarrouk, Siegfried Handschuh, and Brian Davis. Semeval-2017 task 5: Fine-grained sentiment analysis on financial microblogs and news. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 519–535, 2017.

[44] Seunghak Yu, Giovanni Da San Martino, and Preslav Nakov. Experiments in detecting persuasion techniques in the news. *arXiv preprint arXiv:1911.06815*, 2019.

[45] Giovanni Da San Martino, Alberto Barrón-Cedeño, Henning Wachsmuth, Rostislav Petrov, and Preslav Nakov. Semeval-2020 task 11: Detection of propaganda techniques in news articles. In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 1377–1414, 2020.

[46] Dimitar Dimitrov, Bishr Bin Ali, Shaden Shaar, Firoj Alam, Fabrizio Silvestri, Hamed Firooz, Preslav Nakov, and Giovanni Da San Martino. SemEval-2021 task 6: Detection of persuasion techniques in texts and images. In *Proceedings of the 15th International Workshop on Semantic Evaluation (SemEval-2021)*, pages 70–98, Online, August 2021. Association for Computational Linguistics.

[47] Dawid Jurkiewicz, Łukasz Borchmann, Izabela Kosmala, and Filip Graliński. ApplicaAI at SemEval-2020 task 11: On RoBERTa-CRF, span CLS and whether self-training helps them. In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 1415–1424, Barcelona (online), December 2020. International Committee for Computational Linguistics.

[48] Kyle Hamilton. Towards an ontology for propaganda detection in news articles. In *The Semantic Web: ESWC 2021 Satellite Events: Virtual Event, June 6–10, 2021, Revised Selected Papers 18*, pages 230–241. Springer, 2021.

[49] Giovanni Da San Martino, Shaden Shaar, Yifan Zhang, Seunghak Yu, Alberto Barrón-Cedeño, and Preslav Nakov. Prta: A system to support the analysis of propaganda techniques in the news. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 287–293, Online, July 2020. Association for Computational Linguistics.

[50] Nikolaos Nikolaidis, Nicolas Stefanovitch, and Jakub Piskorski. On experiments of detecting persuasion techniques in Polish and Russian online news: Preliminary study. In *Proceedings of the 9th Workshop on Slavic Natural Language Processing 2023 (SlavicNLP 2023)*, pages 155–164, Dubrovnik, Croatia, May 2023. Association for Computational Linguistics.

[51] Joel Young, Craig Martell, Pranav Anand, Pedro Ortiz, Henry Tucker Gilbert IV, et al. A microtext corpus for persuasion detection in dialog. In *Workshops at the Twenty-Fifth AAAI Conference on Artificial Intelligence*. Citeseer, 2011.

[52] Mesut Erhan Unal, Adriana Kovashka, Wen-Ting Chung, and Yu-Ru Lin. Visual persuasion in covid-19 social media content: A multi-modal characterization. In *Companion Proceedings of the Web Conference 2022*, pages 694–704, 2022.

[53] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[54] Graham Neubig. Neural machine translation and sequence-to-sequence models: A tutorial, 2017.

[55] Virginia Teller. Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition daniel jurafsky and james h. martin (university of colorado, boulder) upper saddle river, nj: Prentice hall (prentice hall series in artificial intelligence, edited by stuart russell and peter norvig), 2000, xxvi+934 pp; hardbound, isbn 0-13-095069-6, $64.00. *Computational Linguistics*, 26(4):638–641, 2000.

[56] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An Introduction to Statistical Learning: With Applications in R*. Springer Publishing Company, Incorporated, 2014.

[57] Zellig S. Harris. Distributional structure. *WORD*, 10(2-3):146–162, 1954.

[58] Daniel Jurafsky and James H. Martin. *Speech and Language Processing (2nd Edition)*. Prentice-Hall, Inc., USA, 2009.

[59] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An introduction to statistical learning*, volume 112. Springer, 2013.

[60] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[61] Bernhard E Boser, Isabelle M Guyon, and Vladimir N Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152, 1992.

[62] George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

[63] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155, 2003.

[64] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 3111–3119. Curran Associates, Inc., 2013.

[65] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[66] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

[67] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017.

[68] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. *Learning Representations by Back-Propagating Errors*, page 696–699. MIT Press, Cambridge, MA, USA, 1988.

[69] Paul J Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990.

[70] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[71] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[72] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[73] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate, 2014.

[74] Sebastian Raschka, Yuxi Hayden Liu, Vahid Mirjalili, and Dmytro Dzhulgakov. *Machine Learning with PyTorch and Scikit-Learn: Develop machine learning and deep learning models with Python.* Packt Publishing Ltd, 2022.

[75] Thang Luong, Hieu Pham, and Christopher D. Manning. Effective approaches to attention-based neural machine translation. *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 2015.

[76] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

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

[78] Stephan Raaijmakers. *Deep Learning for Natural Language Processing.* Simon and Schuster, 2022.

[79] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.

[80] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[81] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[82] TB Brown, B Mann, N Ryder, M Subbiah, J Kaplan, P Dhariwal, A Neelakantan, P Shyam, G Sastry, A Askell, et al. Language models are few-shot learners. arxiv 2020. *arXiv preprint arXiv:2005.14165*.

[83] Kevin Clark, Minh-Thang Luong, Quoc V Le, and Christopher D Manning. Electra: Pre-training text encoders as discriminators rather than generators. *arXiv preprint arXiv:2003.10555*, 2020.

[84] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in neural information processing systems*, pages 5753–5763, 2019.

[85] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.

[86] Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. Unsupervised cross-lingual representation learning at scale. *arXiv preprint arXiv:1911.02116*, 2019.

[87] Michal Marcinczuk, Jan Kocon, and Maciej Janicki. Liner2 - a customizable framework for proper names recognition for polish. In Robert Bembenik, Lukasz Skonieczny, Henryk Rybinski, Marzena Kryszkiewicz, and Marek Niezgodka, editors, *Intelligent Tools for Building a Scientific Information Platform*, volume 467 of *Studies in Computational Intelligence*, pages 231–253. Springer, 2013.

[88] Quoc V. Le and Tomas Mikolov. Distributed representations of sentences and documents. In *ICML*, volume 32 of *JMLR Workshop and Conference Proceedings*, pages 1188–1196. JMLR.org, 2014.

[89] Hiroki Nakayama, Takahiro Kubo, Junya Kamura, Yasufumi Taniguchi, and Xu Liang. doccano: Text annotation tool for human, 2018. Software available from https://github.com/doccano/doccano.

[90] Vikas Bhardwaj, Rebecca Passonneau, Ansaf Salleb-Aouissi, and Nancy Ide. Anveshan: A framework for analysis of multiple annotators' labeling behavior. In *Proceedings of the Fourth Linguistic Annotation Workshop*, pages 47–55, Uppsala, Sweden, July 2010. Association for Computational Linguistics.

[91] Gabriel Domingos de Arruda, Norton Trevisan Roman, and Ana Maria Monteiro. An annotated corpus for sentiment analysis in political news. In *Anais do X Simpósio Brasileiro de Tecnologia da Informação e da Linguagem Humana*, pages 101–110. SBC, 2015.

[92] Maria Pontiki, Dimitris Galanis, John Pavlopoulos, Harris Papageorgiou, Ion Androutsopoulos, and Suresh Manandhar. SemEval-2014 task 4: Aspect based sentiment analysis. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 27–35, Dublin, Ireland, August 2014. Association for Computational Linguistics.

[93] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[94] Duyu Tang, Bing Qin, Xiaocheng Feng, and Ting Liu. Effective lstms for target-dependent sentiment classification. *arXiv preprint arXiv:1512.01100*, 2015.

[95] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, R'emi Louf, Morgan Funtowicz, and Jamie Brew. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771, 2019.

[96] Fabio Souza, Rodrigo Nogueira, and Roberto Lotufo. Portuguese named entity recognition using bert-crf. *arXiv preprint arXiv:1909.10649*, 2019.

[97] Zhengjie Gao, Ao Feng, Xinyu Song, and Xi Wu. Target-dependent sentiment classification with bert. *IEEE Access*, 7:154290–154299, 2019.

[98] Youwei Song, Jiahai Wang, Zhiwei Liang, Zhiyue Liu, and Tao Jiang. Utilizing bert intermediate layers for aspect based sentiment analysis and natural language inference. *arXiv preprint arXiv:2002.04815*, 2020.

[99] Ikuya Yamada, Akari Asai, Jin Sakuma, Hiroyuki Shindo, Hideaki Takeda, Yoshiyasu Takefuji, and Yuji Matsumoto. Wikipedia2Vec: An efficient toolkit for learning and visualizing the embeddings of words and entities from Wikipedia. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 23–30. Association for Computational Linguistics, 2020.

[100] Jakub Piskorski, Nicolas Stefanovitch, Giovanni Da San Martino, and Preslav Nakov. Semeval-2023 task 3: Detecting the category, the framing, and the persuasion techniques in online news in a multi-lingual setup. In *Proceedings of the 17th International Workshop on Semantic Evaluation*, SemEval 2023, Toronto, Canada, July 2023.

[101] Johannes Bjerva. Will my auxiliary tagging task help? estimating auxiliary tasks effectivity in multi-task learning. In *Proceedings of the 21st Nordic Conference on Computational Linguistics*, pages 216–220, 2017.

[102] T Zhang, X Gong, and CLP Chen. Bmt-net: Broad multitask transformer network for sentiment analysis. *IEEE Transactions on Cybernetics*, 52(7):6232–6243, 2022.

[103] Ruidan He, Wee Sun Lee, Hwee Tou Ng, and Daniel Dahlmeier. An interactive multi-task learning network for end-to-end aspect-based sentiment analysis. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 504–515, Florence, Italy, July 2019. Association for Computational Linguistics.

[104] Victor Sanh, Thomas Wolf, and Sebastian Ruder. A hierarchical multi-task approach for learning embeddings from semantic tasks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6949–6956, 2019.

[105] Xinyi Wang, Guangluan Xu, Zequn Zhang, Li Jin, and Xian Sun. End-to-end aspect-based sentiment analysis with hierarchical multi-task learning. *Neurocomputing*, 455:178–188, 2021.

[106] Clayton Hutto and Eric Gilbert. Vader: A parsimonious rule-based model for sentiment analysis of social media text. In *Proceedings of the international AAAI conference on web and social media*, volume 8, pages 216–225, 2014.