FACULTY
OF APPLIED SCIENCES
UNIVERSITY
OF WEST BOHEMIA

University of West Bohemia
Department of Computer Science and Engineering
Univerzitní 8
30614 Plzeň
Czech Republic

Multilingual Sentiment Analysis

The State of the Art and the Concept of Ph.D. Thesis

Pavel Přibáň

Technical Report No. DCSE/TR-2020-07
August 2020
Distribution: public

Technical Report No. DCSE/TR-2020-07
August 2020

Multilingual Sentiment Analysis

The State of the Art and the Concept of Ph.D. Thesis

Pavel Přibáň

Abstract

Natural language processing (NLP) became an essential part of the artificial intelligence field that is used daily in industry and by millions of people. Sentiment analysis as a part of NLP is no exception. Most of research in sentiment analysis has been done primarily for English, creating a significant gap in performance between English and other languages.
In this thesis, we describe the fundamental theory behind sentiment analysis. We summarize tasks of sentiment analysis and tasks that are related to sentiment analysis. Along with a description of common basic and initial approaches, we also cover recent state-of-the-art techniques for sentiment analysis.
We provide a detailed description of common machine learning techniques used for sentiment analysis and also very recent state-of-the-art machine learning methods and architectures of neural networks and deep learning. Further, we describe cross-lingual approaches that allow a knowledge transfer between languages.
In the end, we recap our preliminary work and future direction and the aims of the final doctoral thesis.

This work was supported by Grant No. SGS-2019-018 Processing of heterogeneous data and its specialized applications.

Contents

Chapter 1

Introduction

In recent years, we have seen a growing interest in Natural Language Processing (NLP) and its applications, Sentiment Analysis (SA) is no exception. Sentiment analysis is not a single task; it is rather a set of several tasks related to discovering opinion, sentiment, emotion, attitude and other subjective information in a text.

SA is used daily by companies allowing them to understand the opinion of customers about their products, services or the company itself. Such information is precious for any company. Another usage is in recommendation systems[1] where the system can customize or recommend any content (ads, movies, songs, products etc.) to its users. The recommendations are based on the user's previous interactions with the system, for example, comments, reviews or ratings of the content.

SA can be divided into different sub-tasks like aspect-based SA, polarity or fine-grained SA or emotion detection. SA can also be applied on many different levels of scope – document-level, sentence or phrase level. A typical example of an SA task is a text polarity detection, in which the goal is to assign an overall sentiment of a given text. The first SA research papers at the beginning of the 21st century (2000-2012) were mostly focused on online reviews of movies, hotels or e-shop products. With a boom of social networks like Facebook or Twitter, which became extremely popular among the entire population, the researchers also moved their interest to social media content. The initial approaches used traditional machine learning methods such as Naive Bayes, Logistic Regression or Support Vector Machines (SVM) classifiers along with sentiment lexicons[2] or bag-of-words features. In recent years, the state-of-the-art systems use almost exclusively neural networks and deep learning techniques.

Despite the years of research, real applications for SA are still challenging

[1]Such system can be used in any social media network, e-shop or movie/music streaming service.

[2]Sentiment lexicon is a list of words or phrases with their corresponding sentiment orientation.

because of several aspects, such as multilinguality and domain dependence. Another aspect that makes SA difficult is that most of the available datasets are annotated for English texts and low-resourced languages suffer from a lack of annotated datasets on which machine learning models could be trained. The issue with low-resourced languages, which is common for the majority of NLP tasks, could be tackled by cross-lingual techniques. A cross-lingual system is able to learn a model from resource-rich language and apply it to the low-resourced language or languages.

This thesis aims to overview SA methods and their corresponding techniques and tools. Further, the thesis focuses on multilingual and cross-lingual approaches.

The thesis is organized as follows: the SA tasks are stated in Chapter 2. Chapter 3 describes conventional machine learning algorithms, including recent deep learning architectures. The methods for text meaning representation are mentioned in Chapter 4. Applicable NLP approaches for SA (including state-of-the-art methods) are discussed in Chapter 5. Chapter 6 contains approaches and techniques for multilingual SA and cross-lingual techniques for knowledge transfer between languages. Chapter 8 summarizes preliminary ideas for future work and goals of the doctoral thesis.

Chapter 2

Sentiment Analysis

The task of sentiment analysis aims at detection, understanding and extraction of subjective information (e.g., opinions, sentiments and emotions) expressed in a text [Liu et al., 2010]. SA is one of the essential downstream tasks in NLP. At the beginning of the 21st century[1], it has become one of the fastest-growing research areas in NLP [Mäntylä et al., 2018]. Companies use it daily to find out whether their customers have a positive, neutral or negative opinion towards their products or services.

2.1 Tasks Overview

Generally, *sentiment analysis* or *opinion mining* can be seen as a collection of distinct tasks related to subjective information extraction and other sub-tasks which are relevant and linked to these tasks. In this section, we summarize, describe and define the most common tasks.

Liu et al. [2010] define and describe several tasks within the field of SA, i.e., polarity detection at *document-level, sentence-level, aspect-based-level* and *comparative SA*. In [Feldman, 2013], the author distinguishes between these four tasks and one extra task called *sentiment lexicon acquisition*, which can also be called *sentiment lexicon generation*. From [Liu, 2012], we can also add *subjectivity classification* and *opinion spam detection*. Further, we briefly describe these tasks and later in this chapter, we define some of them more precisely.

Similar to [Liu, 2012], we use the term *opinion* and *sentiment* interchangeably to denote opinion, sentiment, attitude and emotion, but we have to note that they are not equivalent and we will distinguish them when needed. In general, the research of SA has been conducted mainly for polarity detection task at three levels of granularity – *document-level, sentence-level* and *entity and aspect-based level*. From now, we will refer to these three

[1]Nearly $7,000$ papers related to SA has been published since 2004 [Mäntylä et al., 2018].

types jointly as *sentiment analysis* or *polarity detection* tasks. These tasks can be also considered as a text classification tasks that are usually solved by the conventional supervised machine learning techniques, see Chapter 3. Based on [Pang and Lee, 2008, Liu et al., 2010, Liu, 2012, Feldman, 2013, Medhat et al., 2014], we summarize the tasks as follows:

1. **Polarity detection:** In most cases, the goal is to detect a sentiment polarity (*positive, negative, neutral*) that is expressed towards a given target. The polarity can also be defined with a different number of labels, i.e., *very positive, positive, neutral, negative, very negative*, which is usually referred as *fine-grained sentiment analysis.* Another possibility is to define only *positive, negative* labels, which will result in a binary text classification problem.

    - **Document-level:** The task at this level is to assign an overall sentiment polarity to a given document. For example, given a short Twitter text "*I love the new Zombieland movie #cinema*" which is a review about a particular movie written by a user, the task is to decide whether the user likes (positive sentiment) or dislikes (negative sentiment) the movie. In this task, we assume that the document contains only one opinion towards one entity.
    - **Sentence-level:** This task is almost identical to the *document-level* task, but it is performed on sentences instead of documents. The goal is to classify whether a sentence expresses a positive, negative or neutral sentiment. Again, we assume that the sentence contains only a single sentiment (opinion).
    - **Entity and Aspect-Based level:** The *aspect-based* task[2] evaluates sentiments of individual entities and/or their aspects. Consider the following review of a hotel "*The room was very comfortable and the breakfast was great.*". There are two aspects of the hotel – *room* and *breakfast*, both of them are positive. This task allows to evaluate sentiment in a text (document, sentence) with multiple sentiments and multiple entities and their aspects.

2. **Emotion Detection (Analysis):** In the *emotion detection* task, the system intended for this task must detect a person's emotion expressed in a text. Emotions represent subjective feelings and thoughts of human beings.

3. **Subjectivity Classification:** The sentence-level SA can be done only on sentences with the sentiment, opinion or subjective views. This is the goal of the *subjectivity classification* task [Wiebe et al., 1999, Wiebe

[2]In this context, the word *aspect* can be used interchangeably with word *feature*, thus the task is also called *feature-based sentiment analysis.*

and Riloff, 2005], i.e., detect *objective sentences* that express factual information and *subjective sentences* that express subjective views and opinions. However, even a subjective sentence may not express any sentiment, see Section 2.5.

4. **Comparative Sentiment Analysis:** Commonly, the sentiment is not expressed directly, instead the comparison is used, for example, "*Apple iPhone Xs is much more reliable than Samsung Galaxy S9*", such sentence contains *comparative opinion*. The goal is to identify sentences that contain comparative opinions, extract the comparative opinions expressed in the sentences and select the preferred entities (Apple iPhone Xs, in our example).

5. **Other:** Other tasks related to SA like *Opinion Spam Detection, Sentiment Lexicon Acquisition (Generation), Sarcasm Detection (Analysis), Opinion Summarization* and others. We describe some of them in sections 2.7.1, 2.7.2, 2.7.3.

For the next part of this thesis, we have to define and explain the opinion and other related terms.

2.2 Opinion Definition

We define *opinion* for the SA task according to [Liu, 2012] as a quadruple:

$$(g, s, h, t) \tag{2.1}$$

where g is an opinion target, s is an opinion polarity (sentiment), h is an opinion holder and t is a time when the opinion was expressed.

For the explanation of components from the previous definition, we will use a similar example to the one stated in [Liu, 2012]. The example is:

Author: Nick Newman, 25/10/2019

"(1) *I really like my new Samsung TV.* (2) *I cannot live without it.* (3) *The resolution is unbelievable.* (4) *But the price is not so good as the resolution.* (5) *My friends love it too.* (6) *This Samsung TV is definitely better than my old Philips TV.*"

The opinion target g can be any entity or aspect of the entity about which the opinion has been expressed. For example, in the sentence (1) the target of the opinion is *Samsung TV* with positive sentiment. The example of a target which is an aspect, is in the sentence (3), the target is *resolution.*

Secondly, the example contains opinions of two entities. Sentences (1), (2), (3), (4) are opinions of the author of the review (*Nick Newman*) and in the sentence (5) expressed an opinion of the author's friends. These entities are referred as *opinion sources* or *opinion holders* [Kim and Hovy, 2004, Wiebe et al., 2005]. Lastly, the date of the example is 25/10/2019 and the reason why the opinion definition contains time t (or date) is that the sentiment can evolve during a time and it is useful to observe these changes over time.

2.2.1 Entity Definition

Next, we define the term ***entity*** as the target object that has been evaluated. An ***entity e*** can be a product, service, topic, issue, person, organization or event. Formally it is defined as a pair $e{:}(T, W)$, where T is a hierarchy of *parts* (or *components*) and *sub-parts* of the entity and W is a set of *attributes* of e. For example, one part of the *Samsung TV* is a screen which is composed of other sub-parts like screen glass, LED display, frame etc. The root node is the entity itself (*Samsung TV*) and other nodes contain parts and sub-parts. Each part or sub-part has its own set of attributes, for example, a resolution is an attribute of the LED display.

An opinion can be expressed on any node or on any attribute of the node. In the previous example in the sentence (1) author expressed the opinion on the *Samsung TV* itself (root node) and in the sentence (2), he expressed his opinion on one of its attributes (*resolution*).

This hierarchical description of an entity with any number of levels and nested relations is universal but often too complex for some real applications. The difficulty of applying SA for such a universal hierarchical definition is though and challenging. Thus, we simplify the hierarchy according to [Liu et al., 2010] to two levels and use the term ***aspects*** to denote both parts (sub-parts) and attributes, see Figure 2.1.

After the simplification, the root of the node is still the entity[3] itself and the other nodes are aspects of the entity.

2.2.2 Entity and Aspect-Based level Sentiment Analysis

The previous definition of opinion in Section 2.2 was sufficient for text unit (document, sentence, paragraph) with one opinion towards one entity. In the case of aspect-based level SA, the task is to discover all or multiple opinions towards individual entities and/or their aspects in a given opinion document d. Thus, we extend the previous definition of opinion with the entity

[3]Entity is sometimes also called *object* and aspects can be also called *features, facets, attributes* or *topics*.

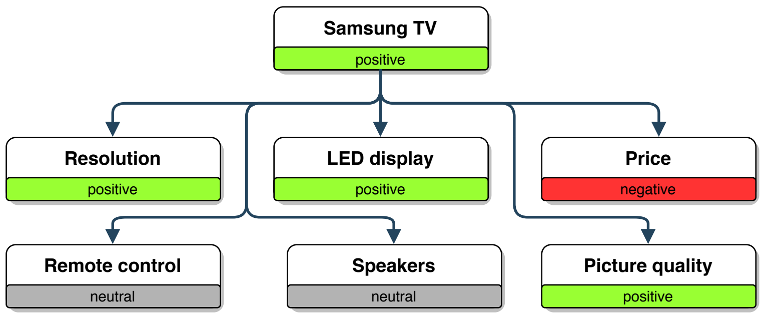


Figure 2.1: Simplified example of hierarchical representation of an entity (*Samsung TV* entity).

definition from the previous Section according to [Liu, 2012] as quintuple:

$$(e_i, a_{ij}, s_{ijkl}, h_k, t_l) \tag{2.2}$$

where e_i is a name of an entity, a_{ij} is a *j-th* aspect of entity e_i, h_k is an opinion holder, t_l is the time when the opinion was expressed and s_{ijkl} is a sentiment on aspect a_{ij} of entity e_i in time t_l expressed by opinion holder h_k. In the case where the overall opinion is expressed towards the entity itself special aspect named *GENERAL* is used to denote it. The e_i and a_{ij} pair substitute the target g from the definition 2.1.

The opinion document d (or other unit of text like paragraph, sentence) is then composed from a set of opinion quintuples $O = \{o_1, o_2, \ldots, o_m\}$ expressed on a set of entities $E = \{e_1, e_2, \ldots, e_r\}$ and their aspects with a set of opinion holders $H = \{h_1, h_2, \ldots, h_p\}$ at some certain time point. An entity e_i is represented by itself and by a set of aspects $A_i = \{a_{i1}, a_{i2}, \ldots, a_{in}\}$.

The aspect of a particular entity can be either *implicit* or *explicit*. The explicit aspects are usually expressed with nouns or noun phrases, for example, "*The resolution of the Samsung TV is impressive.*" is the explicit aspect. The implicit aspects are usually expressed with adverbs, adjectives or even verbs. For example, the sentence "*This Samsung TV is really expensive.*" implies that there is an aspect *price* with negative sentiment, although the price was not explicitly mentioned in the sentence.

Both definitions of opinions (2.1 and 2.2) are not able to handle all possible options and cases in which opinion can be expressed, but they are sufficient for most applications. Examples in which these definitions fail are shown in [Liu, 2012].

The whole task of aspect-based SA, in other words obtaining the entire set O of opinions for certain document d, is composed of several sub-tasks like entity extraction, aspect extraction and categorization, time extraction and etc.

2.2.3 Opinion Types

In the previous sections, we described only one type of opinion, which is called *regular opinion* (definition 2.2), but there are two main types of opinions – ***regular opinion*** and ***comparative opinion***.

- **Regular opinion** or just *opinion* can be divided into two categories [Liu, 2006]:
  - **Direct opinion** is expressed directly towards an entity or its aspect, for example, "The resolution is unbelievable."
  - **Indirect opinion** is expressed indirectly on an entity or its aspect on some other entities. For example, "*Once I finished the lunch I had a stomachache and I was vomiting the whole day.*" implies that the food was spoiled and the person (the other entity) vomited, which implies negative opinion towards the food.
- **Comparative opinion** expresses a relation of similarities or differences between two or more entities and/or a preference of the opinion holder based on some of the shared aspects of the entities [Liu, 2006, Jindal and Liu, 2006a,b]. For example, the sentence (6) from the example at the beginning of this Section 2.2, contains comparative opinion. Comparisons can be divided into two main groups ***gradable comparison*** and ***non-gradable comparison*** which are described in more detail in [Liu, 2012] and [Liu, 2006], and partly in Section 2.6.

Next, we recognize ***explicit opinion*** and ***implicit opinion*** which are defined according to [Liu, 2012] as follows:

- **Explicit opinion** is a ***subjective statement*** that gives a regular or comparative opinion, for example, "*I really like my new Samsung TV.*" or "*This Samsung TV is definitely better than my old Philips TV.*"
- **Implicit opinion** is an ***objective statement*** that implies a regular or comparative opinion. For example, "*My new Samsung TV has stopped working after a few days*" or "*The resolution of my new Samsung TV is higher than my old Philips TV.*" Implicit opinions often express some desirable or undesirable features, defects, properties, attributes or consequences for target entities or their aspects.

2.2.4 Author and Reader Standing Point

The opinion can be considered from two perspectives, the author of the opinion (opinion holder) or the reader of the opinion. Thanks to this assumption, one sentence can be negative as well as positive. For example, in a sentence:

"Our national team lost against Germany which is really bad" the author expresses the negative opinion about the loss against Germany, but for a reader which is a fan of Germany team it holds the positive sentiment. Usually, the opinion holders are assumed to be the consumers unless otherwise specified [Liu, 2012].

2.3 Polarity Detection

In this section, we summarize the three primary SA tasks (document-level, sentence-level and aspect-based level) from the overview in Section 2.1. These task aims to a polarity detection[4].

2.3.1 Document-Level

The goal of this task is to assign an overall sentiment polarity s for a given opinion document d expressed towards a given entity by some opinion holder at some time. According to [Liu, 2012] the task is to extract an opinion given by the quintuple defined in 2.2 with aspect GENERAL in the following way:

$$(\_, GENERAL, s, \_, \_)$$

assuming that the entity e, opinion holder h and time of the opinion t are known or irrelevant. This definition also assumes that the opinion expressed in document d is aiming only on one entity e (if known) and there is only one opinion holder h.

Because of this assumption and because the aspect is always GENERAL, we can use the simpler definition of opinion given by the quadruple defined in 2.1 and redefine the task as obtaining only the overall sentiment s for a given document d. Then, the quadruple looks as follows:

$$(\_, s, \_, \_)$$

and again assuming that g, h and t are known or irrelevant.

2.3.2 Sentence-Level

The *sentence-level* polarity detection aims to a detection of sentiment in a sentence. In this task, we still assume that the sentence contains only one opinion towards one entity[5]. One sentence can be considered as a single document, thus similar or identical approaches to *document-Level* task can be applied. The *sentence-level* polarity detection task can be used for longer documents in which each sentence is evaluated independently as one

[4]The *polarity detection* task can be also referred as *sentiment analysis* task.

[5]Despite the fact that this assumption is incorrect in many examples.

document, which results in a set of sentences with assigned sentiment. Alternatively, the assigned sentiments can be summarized to represent the overall sentiment of the document.

Liu [2012] defines the *sentence-level* polarity detection as follows: given a sentence x, determine whether x expresses a positive, negative or neutral opinion. If there is no opinion the sentence is considered as neutral.

He also mentions two possible ways how to solve this problem. The first one is to represent the task as a classical three-class classification problem. In the second approach, he first decided whether the sentence expresses an opinion or not, which is the *subjectivity classification* task. The second step is then to classify the subjective sentence into a positive or negative class. The first step can be problematic for some sentences because, as we mentioned, even the objective sentence can imply opinions.

For most cases in practice, the *sentence-level* and *document-Level* polarity detection is suitable for short reviews, Twitter or Facebook posts or other short text with one or few sentences expressing one opinion towards one entity.

2.3.3 Entity and Aspect-based Level

The two previously described tasks were focused only on the overall sentiment of one entity in the entire document or sentence. Commonly, documents and sentences often contain more than one opinion (sentiment) expressed towards an entity (eventually multiple entities) or its aspects. At typical example where aspect-based level SA fits perfectly is analysis of product reviews. In the product reviews, people usually express opinions towards aspects or attributes of the product and thus detecting individual sentiment (opinion) towards each aspect is desirable.

The complete definition of the task is introduced in 2.2.2. To recall, the goal of this task is to obtain all quintuples $(e_i, a_{ij}, s_{ijkl}, h_k, t_l)$ for document d, where e_i is a name of an entity, a_{ij} is a *j-th* aspect of entity e_i, h_k is an opinion holder, t_l is the time when the opinion was expressed and s_{ijkl} is a sentiment on aspect a_{ij} of entity e_i in time t_l expressed by opinion holder h_k. In case, where the overall opinion is expressed towards the entity itself, a special aspect named *GENERAL* is used to denote it. In practice, some members of the quintuple 2.2 can be omitted because they are unimportant, irrelevant or known. Aspect-based level SA is a complex task which consists of several sub-tasks [Liu, 2006, Liu et al., 2010, Liu, 2012].

1. **Entity extraction and categorization:** Find and extract all mentions and synonyms of entities in a given document d and assign them corresponding category. Each category then represents one entity e_i from a set of entities $E = \{e_1, e_2, \ldots, e_r\}$.

2. **Aspect extraction and categorization:** Find and extract all aspect expressions for all entities obtained in the first task and classify the aspect expressions. Each entity e_i has its own set of aspects (categories) $A_i = \{a_{i1}, a_{i2}, \ldots, a_{in}\}$ where a_{ij} represents one unique aspect of entity e_i.

3. **Opinion holder extraction and categorization:** Find and extract opinion holders or their mentions from text or structured data and assign them corresponding category. The output of this task is a set of opinion holders $H = \{h_1, h_2, \ldots, h_p\}$.

4. **Time extraction and standardization:** Extract the times when opinions were expressed and standardize different time formats.

5. **Aspect sentiment classification:** Classify an opinion on aspect a_{ij} as positive, negative or neutral or assign other predefined sentiment classes.

6. **Opinion quintuple generation:** For document d, create all opinion quintuples $(e_i, a_{ij}, s_{ijkl}, h_k, t_l)$, with results from previous tasks.

2.3.4 Corpora for Sentiment Analysis

Datasets are required not only to train the supervised machine learning models but also to evaluate the performance of any system or approach. In this section, we summarize some of the well known and popular English datasets for the polarity detection task. We also include examples of datasets for other languages, even though most of the research was focused on English. The datasets for *sentence-level* and *document-level* tasks are listed in Table 2.1 and datasets for *aspect-based* task are shown in Table 2.2.

Manual annotation of datasets for NLP tasks is usually very expensive and time-consuming therefore, researchers are trying to find approaches to get labeled data in another way. In SA, there are two main approaches to obtain labeled data: (1) *manual annotation* and (2) *distant supervision* [Giachanou and Crestani, 2016].

The manual annotation for simpler tasks like polarity detection can be done in any spreadsheet (MS Excel, Google Sheets etc.) or software specialized for data annotating. The manual annotation is usually applied for more complex tasks like aspect-based SA. Examples of manually annotated datasets can be found in [Socher et al., 2013, Habernal et al., 2013, Dong et al., 2014, Saeidi et al., 2016] or [Rosenthal et al., 2017]. The advantage of the manual annotation is that the labels are more reliable and less erroneous.

The second option is the *distant supervision* approach. It automatically allows to label data with a minimal human interaction or completely without incorporating human into the annotating process. The distant supervision techniques use some metadata or some specific property of the data to obtain the label. There are two main sources for distant supervision datasets. The first one is review of any type. The textual reviews usually contain also some additional explicit rating (e.g., number of stars), this explicit rating is used as a label for the textual review, such approach was used, for example, in [Pang et al., 2002, Pang and Lee, 2004, Maas et al., 2011] or [Habernal et al., 2013]. The second source is social media websites (e.g., Twitter, Facebook). In this case, predefined emoticons, emojis or hashtags are used as a noisy label. The predefined emoticons or hashtags are tied up with a certain label (class) and based on their presence in the post, the corresponding class is used as the label. For example, in Tweet *"I'm so happy :) #cool #amazing"* emoticon *":)"* the hashtag *"#amazing"* is assigned to the positive sentiment and thus the Tweet is labeled as positive. Similar method was used in [Go et al., 2009] and [Speriosu et al., 2011]. With this approach, a huge amount of annotated data can be obtained, but the reliability is lower compared to the manual approach. Another disadvantage of Twitter datasets is that they cannot be redistributed because of the Twitter license, only IDs of tweets can be published and everyone has to download the tweets by himself. It is not unusual that some tweets are deleted or privatized and the original dataset cannot be reconstructed anymore.

The *SST-2* dataset [Socher et al., 2013] is part of the popular *General Language Understanding Evaluation* (GLUE) [Wang et al., 2019] benchmark. The GLUE benchmark is a collection of resources for training, evaluating and analyzing natural language systems in a diverse set of existing NLP tasks.

Paper	Name	Size	Classes	Text	Domain	Source	Annotation	Language
[Pang et al., 2002]	Movie Reviews v1.0	1,400	P, N	review	movie reviews	IMDb	ratings	English
[Pang and Lee, 2004]	Movie Reviews v2.0	2,000	P, N	review	movie reviews	IMDb	ratings	English
[Pang and Lee, 2005]	Sentence Polarity	10,662	P, N	sentence	movie reviews	Rott. Tom.	ratings	English
[Go et al., 2009]	Sentiment140	1,600,000	P, N	tweet	multiple	Twitter	emoticons	English
[Go et al., 2009]	Sentiment140 Test	359	P, N	tweet	multiple	Twitter	manual	English
[Shamma et al., 2009]	Obama-McCain Debate	1,904	P, N	tweet	Obama McCain	Twitter	manual	English
[Maas et al., 2011]	IMDb	50,000	P, N	review	movie reviews	IMDb	ratings	English
[Socher et al., 2013]	SST-5	11,855	P+, N+, O	sentence	movie reviews	Rott. Tom.	manual	English
[Socher et al., 2013]	SST-2	9,613	P, N	sentence	movie reviews	Rott. Tom.	manual	English
[Zhang et al., 2015b]	Yelp-Fine	140,000	P+, N+, O	review	multiple	Yelp	ratings	English
[Zhang et al., 2015b]	Yelp-Binary	299,000	P, N	review	multiple	Yelp	ratings	English
[Rosenthal et al., 2017]	SemEval-2017	62,617	P, N, O	tweet	multiple	Twitter	manual	English
[Speriosu et al., 2011]	Health Care Reform	2,394	P, N, O	tweet	health care tweets	Twitter	emoticons	English
[Habernal et al., 2013]	Czech Social Media	10,000	P, N, O	FB post	multiple	Facebook	manual	Czech
[Habernal et al., 2013]	Czech Movie Reviews	91,381	P, N, O	review	movie reviews	CSFD	ratings	Czech
[Habernal et al., 2013]	Czech Prodcut Reviews	145,307	P, N, O	review	product reviews	MALL.cz	ratings	Czech
[Villena-Román, 2013]	TASS-2013	68,017	P+, N+, O, X	tweet	multiple	Twitter	mixed	Spanish
[Barbieri et al., 2016]	Evalita-2016	9,410	P, N, O, M	tweet	multiple	Twitter	mixed	Italian
[Rosenthal et al., 2017]	SemEval-2017-AR	9,455	P, N, O	tweet	multiple	Twitter	manual	Arabic

Table 2.1: Overview of datasets for sentiment polarity classification. Values in column *Size* denotes the number of examples in the dataset. Values in column *Classes* refer to the following classes: [P]: *positive*, [N]: *negative*, [P+]: *very positive* and *positive*, [N+]: *very negative* and *negative*, [O]: *neutral*, [M]: *mixed*, [X]: *none*. Values in column *Text* denotes type (granularity) of textual examples in the dataset (*FB post* stands for Facebook post). Values in column *Domain* represent a domain of the text. Values in column *Source* refer to the source web pages where the data comes from: (IMDb: `www.imdb.com`, Rott. Tom.: `www.rottentomatoes.com`, Twitter: `www.twitter.com`, Yelp: `www.yelp.com`, Facebook: `www.facebook.com`, CSFD: `www.csfd.cz`, MALL.cz: `www.mall.cz`). *Annotation* Column denotes the approach used for obtaining labels; *ratings* and *emoticons* values refer to the *distant supervision method*, *mixed* values mean that a combination of manual and distant supervision was used.

Paper	**Name**	**Size**	**Classes**	**Text**	**Domain**	**Source**	**Annotation**	**Language**
[Saeidi et al., 2016]	SentiHood	5,215	P, N	answers	quiestion answering	Yahoo	manual	English
[Pontiki et al., 2014]	SemEval-2014	7,686	P, N, O, C	sentence	laptops and restaurants reviews	-	manual	English
[Pontiki et al., 2016]	SemEval-2016	70,790	P, N, O	sentence	multiple	-	manual	Multilingual
[Dong et al., 2014]	Target Dependent	6,940	P, N, O	tweet	multiple	Twitter	manual	English

Table 2.2: Overview of datasets for aspect-based SA. Values in all columns have the same meaning as in Table 2.1. The letter *C* in column *Classes* refers to the *conflict* class and the *Yahoo* string refers to the `www.yahoo.com` website.

2.4 Emotion Analysis

In this section, we discuss emotions and tasks related to them. Emotions are our subjective feelings and thoughts and they can be perceived or expressed with different levels of intensity[6]. The intensity denotes the degree or quantity of the emotion, for example, *really excited, very happy* or *litle bit angry* etc. [Liu, 2012, Liu et al., 2010, Mohammad et al., 2018, Canales and Martínez-Barco, 2014, Shrivastava et al., 2019, Bostan and Klinger, 2018]. Emotion analysis can have many applications, for example, in e-learning [Rodriguez et al., 2012], suicide prevention [Desmet and Hoste, 2013, Vaassen, 2014] or for a prediction of stock market prices [Bollen et al., 2011].

2.4.1 Categorical Model

Emotions have been studied in different research areas, e.g., psychology, philosophy, sociology and also in a field of natural language processing. Humans are able to perceive many different emotions. According to the *basic emotion model* (also called *categorical model*) [Ekman, 1992, Plutchik, 1980, Parrott, 2001, Frijda, 1988] emotions can be categorized into distinct emotion classes. For example, emotions like *joy, sadness, anger, fear* are considered to be more basic than others, i.e., physiologically, cognitively and in terms of the mechanisms to express these emotions [Mohammad et al., 2018]. The definitions in different publications can slightly vary, but the basic idea remains the same. For example, Parrott [2001] distinguishes six primary emotions, i.e., *love, joy, surprise, anger, sadness* and *fear*, which can be further divided into other sub-categories. Ekman [1992] also recognizes six (but slightly different) basic emotions, i.e., *anger, disgust, fear, joy, surprise* and *sadness.* Plutchik [1980] claims that there are eight basic emotions (in the *Plutchik's wheel of emotions*, see Figure 2.2), i.e., *joy, sadness, anger, fear, trust, disgust, surprise* and *anticipation* (the inner circle), and other more complex emotions are in the outer circles, outer circles are also composed of emotions with a smaller degree of intensity. Each primary emotion has a polar opposite, e.g., anticipation is the opposite of surprise. The *Plutchik's wheel of emotions* can be seen as a hybrid model between the categorical and dimensional models. However, here we treat it as a categorical model because the emotions are expressed discretely and not as a continuous number in the n-dimensional space as it is in the *Valence-Arousal-Dominance* model, see Section 2.4.2. We have to note that *emotions* and *opinions* are not equivalent but they are closely related and they have a significant intersection.

[6]Intensity differs from *arousal* dimension from valence-arousal-dominance model. Arousal represents whether an emotion is calming or exciting.

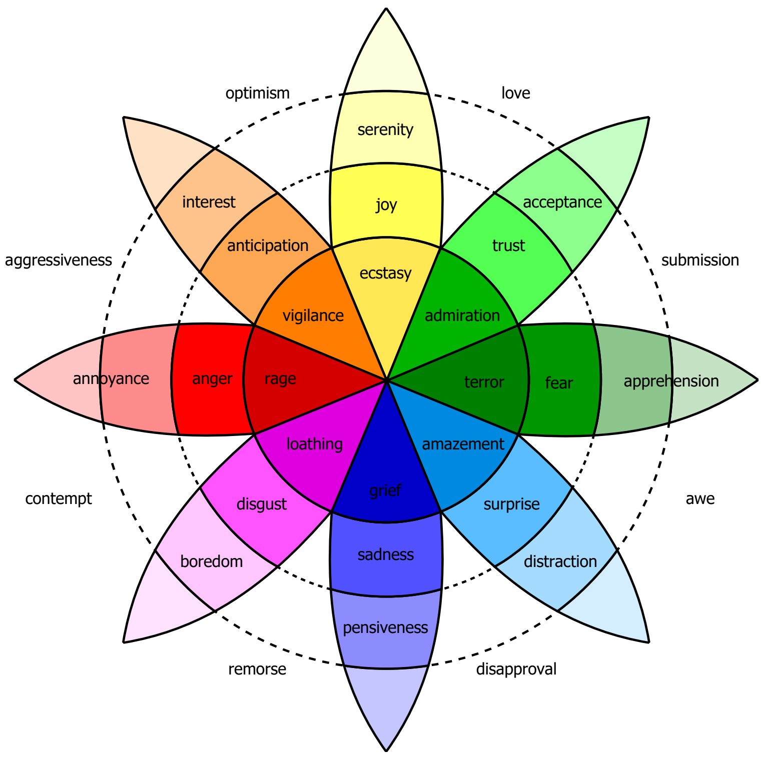


Figure 2.2: Plutchik's wheel of emotions (picture taken from [Commons, 2020]).

2.4.2 Dimensional Model

Dimensional emotion model represents emotions in n-dimensional space. In *Valence-Arousal-Dominance* (VAD) dimensional model, the emotions are points in a three-dimensional space. The model says that there are three largely independent emotional dimensions of word meaning, see Figure 2.3. The *valence* dimension (positiveness-negativeness / pleasure-displeasure) expresses the attractiveness or sentiment of an emotion. The *arousal* dimension (active-passive) represents an activation level of the emotion. The *dominance* dimension (dominant-submissive) represents a level of control over the emotion [Mäntylä et al., 2016, Mohammad, 2018, Osgood et al., 1957, Russell, 1980, 2003]. For example, the word *birthday* indicates more positiveness than the word *death*; *nervous* indicates more arousal than *lazy*; and *fight* indicates more dominance than *fragile*.

The approaches (systems/algorithms) for emotion analysis usually use the categorical models because of its simplicity, the emotions can be categorized in distinct classes or categories. The disadvantage is that the categorical models contain limited number of emotions and may not adequately cover all emotions. The advantage is that it can be used for measuring the similarity between emotions [Canales and Martínez-Barco, 2014].

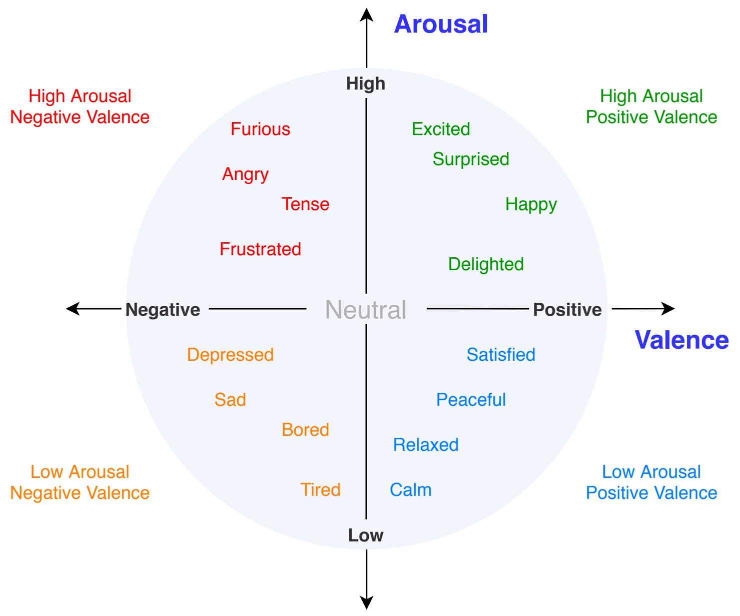


Figure 2.3: Joint visualization of the *Arousal* and *Valence* dimensions with examples of emotions.

2.4.3 Emotion Analysis Tasks

The basic task of *Emotion Analysis* is the ***Emotion Detection*** task, where the goal is to detect various emotions in a given text [Medhat et al., 2014, Liu, 2012, Shrivastava et al., 2019]. Another task is called ***Emotion Intensity Detection*** task. In this task, the intensity of a given text and emotion need to be detected.

This kind of task was investigated in *SemEval-2018 Task 1: Affect in Tweets* shared task [Mohammad et al., 2018]. There were also other shared tasks related to emotion intensity; SemEval-2007 Task 14 [Strapparava and Mihalcea, 2007] and WASSA-2017 shared task on Emotion Intensity [Mohammad and Bravo-Marquez, 2017].

In the shared competition called *Implicit Emotion Shared Task*[7] (IEST) [Klinger et al., 2018] the participants were asked to create a system which should infer one of six emotions (anger, disgust, fear, joy, sadness and surprise) only from a context of a particular emotion word which was removed from the text. For example, "*It's [#TARGETWORD#] when you feel like you are invisible to others.*", the missing word was *sad* and the system should detect *sadness* emotion.

[7]It was a part of *9th Workshop on Computational Approaches to Subjectivity, Sentiment & Social Media Analysis* (i.e., *WASSA 2018*).

2.4.4 Approaches for Emotion Analysis

The approaches for emotion analysis tasks can be divided into *lexicon-based* approach and *machine learning approaches* [Canales and Martínez-Barco, 2014, Buechel and Hahn, 2016]. A straight numerical comparison of available approaches is not possible because different works (papers) usually use different datasets for their evaluations. Tables 2.3 and 2.4 contain an overview of selected papers for the emotion detection task and emotion intensity detection task, respectively. The purpose of these tables is to give the reader a basic overview of the current methods and results in emotion analysis tasks. Datasets from the tables are described in Section 2.4.5, except for the papers where the dataset is not publicly available. Following papers are listed in the tables: [Strapparava and Mihalcea, 2008, Balabantaray et al., 2012, Balahur et al., 2012, Roberts et al., 2012, Buechel and Hahn, 2016, Abdul-Mageed and Ungar, 2017, Baziotis et al., 2018, Huang et al., 2019, Polignano et al., 2019, Agrawal and Suri, 2019, Shrivastava et al., 2019, Köper et al., 2017, Goel et al., 2017] and [Duppada et al., 2018]. Description of other works related to these tasks can be found in [May et al., 2019, Medhat et al., 2014, Canales and Martínez-Barco, 2014, Avetisyan et al., 2016, Buechel and Hahn, 2016]. Next, we briefly describe some works from Tables 2.3 and 2.4.

Paper	**Emotion Model**	**Approach**	**Dataset**	**Result**
[Strapparava and Mihalcea, 2008]	Categorical	LSA	Semeval 2007	18% F_1 Score
[Balabantaray et al., 2012]	Categorical	SVM	Their own	73% Accuracy
[Balahur et al., 2012]	Categorical	SVM	ISEAR	45% F_1 Score
[Roberts et al., 2012]	Categorical	SVM	EmpaTweet	67% F_1 Score
[Buechel and Hahn, 2016]	VAD/Categorical	SVM, kNN	Semeval 2007	0.42 Pearson Correlation
[Abdul-Mageed and Ungar, 2017]	Categorical	GRU	Their own	96% F_1 Score
[Baziotis et al., 2018]	Categorical	LSTM	SemEval 2018	53% F_1 Score†
[Huang et al., 2019]	Categorical	CNN, LSTM	SemEval 2018	65% F_1 Score†
[Polignano et al., 2019]‡	Categorical	CNN, LSTM	SemEval 2018	84% F_1 Micro Score†
[Polignano et al., 2019]‡	Categorical	CNN, LSTM	SemEval 2019	70% F_1 Micro Score
[Agrawal and Suri, 2019]	Categorical	LSTM, Gradient boosting	SemEval 2019	78% F_1 Micro Score
[Polignano et al., 2019]‡	Categorical	CNN, LSTM	ISEAR	52-78% F_1 Score
[Shrivastava et al., 2019]	Categorical	CNN	TV-Charmed	72% F_1 Score

†Polignano et al. [2019] and Huang et al. [2019] used each a different set of emotions for classification than the official participant [Baziotis et al., 2018] of the SemEval 2018 task [Apidianaki et al., 2018] and thus they are not directly comparable even though they use data from the same dataset.
‡ Is the same paper, but they are listed separately for clarity because Polignano et al. [2019] use multiple datasets for evaluation.

Table 2.3: Overview of papers related to the emotion detection task.

Paper	**Approach**	**Dataset**	**Result**
[Köper et al., 2017]	Random Forrest, CNN, LSTM	WASSA 2017	0.72 Pearson Correlation
[Goel et al., 2017]	CNN, LSTM	WASSA 2017	0.74 Pearson Correlation
[Duppada et al., 2018]	XG Boost, Random Forrest	SemEval 2018	0.80 Pearson Correlation
[Huang et al., 2019]	CNN, LSTM	WASSA 2017	0.77 Pearson Correlation

Table 2.4: Overview of papers related to the emotion intensity detection task.

The *lexicon-based* approaches use predefined emotion lexicons (see Section 2.7.1). The initial work was done in Strapparava and Mihalcea [2008], they experimented with a dataset from SemEval 2007 [Strapparava and Mihalcea, 2007]. They use an algorithm that checks a presence of emotion words in headlines and computes a score based on the frequency of those words in the text.

Balahur et al. [2012] combine machine learning and lexicon-based approach. They proposed a method based on a commonsense knowledge base EmotiNet [Balahur et al., 2011]. For evaluation, they used emotion corpus International Survey of Emotional Antecedents and Reactions [Scherer and Wallbott, 1994] (ISEAR) which contains descriptions of real-life situations and one of seven major emotion they usually trigger. [Roberts et al., 2012] downloaded and manually annotated posts from Twitter with one of seven emotions using 14 topics. They tackle emotion detection as a multi-label classification problem. They used seven independent binary Support Vector Machines (SVM) classifiers with similar features to [Balabantaray et al., 2012].

In recent years *deep learning* methods have become very popular in the NLP field and emotion analysis is no exception. The deep learning methods are now used in most cases, for example, 21 out of 26 participating teams in WASSA 2018 IEST task [Klinger et al., 2018] used deep neural network and in SemEval-2019 Task 3: EmoContext [Chatterjee et al., 2019] almost all teams from the top fifteen best performing teams used deep neural network as well.

Huang et al. [2019] used a combination of convolutional neural networks (CNN) and recurrent neural network (RNN) Bidirectional Long-Short-Term memory neural networks [Graves and Schmidhuber, 2005] (BiLSTM) for emotion detection and predicting emotion intensity. Abdul-Mageed and Ungar [2017] applied Gated Recurrent Neural Network [Cho et al., 2014c, Chung et al., 2015].

2.4.5 Emotion Analysis Datasets

In this section, we briefly describe and summarize some datasets for emotion analysis, i.e., for the emotion detection task and the emotion intensity detection task. Overviews of datasets for both tasks are organized in Tables 2.5 and 2.6. Regarding the related work, Bostan and Klinger [2018] summarize and describe other existing and available datasets for emotion detection and map them to a common format in a way that can be used for future research. They also performed cross-domain experiments for emotion detection.

SemEval 2007 dataset [Strapparava and Mihalcea, 2007] was used for the *SemEval-2007 Task 14: Affective Text* shared task at SemEval 2007 [Agirre et al., 2007]. The dataset is composed of $1,250$ annotated news headlines. Each headline annotated with a score from 0 to 100 for each emotion (it is a multi-label annotation). They used six Ekman's [Ekman, 1992] basic emotions. Along with the emotions, the valence of the headline was also annotated with a value from -100 to 100.

Paper	Dataset	Text	Size	Topic	Emotions	Multi
[Strapparava and Mihalcea, 2007]	SemEval 2007	Headlines	1,250	News	E*, R	yes
[Scherer and Wallbott, 1994]	ISEAR	Descriptions	7,667	Events	E*, G, M	no
[Roberts et al., 2012]	EmpaTweet	Tweets	7,000	General	E*, R, L+N	no
[Mohammad et al., 2018]	SemEval 2018	Tweets	10,983	General	E*, P*, R+O, N	yes
[Chatterjee et al., 2019]	SemEval 2019	Dialogues	3,8424	General	A, H, S+O	no
[Shrivastava et al., 2019]	TV-Charmed	Utterances	13,354	TV show	E*, R+O	no
[Schuff et al., 2017]	SSEC	Tweets	4,868	General	E*, R, T, U	yes
[Buechel and Hahn, 2017]	EmoBank	Sentences	10,548	General	-	-

Table 2.5: Overview of datasets for the emotion intensity detection task. Values in column *Multi* denote whether the dataset contains multi-label annotations. Values in column *Emotions* refer to the following emotions and classes: [E*]: *anger, disgust, fear, joy, sadness*, [P*]: *trust, anticipation, love, optimism, pessimism*, [R]: *surprise*, [G]: *guilt*, [M]: *shame*, [L]: *love*, [H]: *happy*, [S]: *sadness*, [T]: *trust*, [U]: anticipation, [A]: *anger*, [O]: *other* or *neutral* class, [N]: *no emotion*.

To the best of our knowledge, **ISEAR** the *International Survey of Emotional Antecedents and Reactions* [Scherer and Wallbott, 1994] corpus is the oldest dataset used for emotion detection in NLP. It contains descriptions of real-life situations. It was built by collecting questionnaires about the real-life situation answered by respondents. The dataset contains 7,667 examples labeled with one of seven emotions.

EmpaTweet dataset[8] [Roberts et al., 2012] is created from Twitter posts that were manually labeled with one of seven emotions (six Ekman's emotions plus *love* emotion) or with no emotion. Each post (Tweet) belongs to one of 14 topics. They obtained and annotated 7,000 Tweets.

Paper	Dataset	Text	Size	Topic	Emotions
[Mohammad et al., 2018]	SemEval 2018	Tweets	12,634	General	anger, fear, joy, sad.
[Mohammad and Bravo-Marquez, 2017]	WASSA 2017	Tweets	7,097	General	anger, fear, joy, sad.

Table 2.6: Overview of datasets for the emotion detection task.

The other datasets stated in the tables are described in [Mohammad et al., 2018, Chatterjee et al., 2019, Mohammad and Bravo-Marquez, 2017, Shrivastava et al., 2019, Buechel and Hahn, 2017, Schuff et al., 2017].

[8]Even though they claim that the dataset is publicly available they do not provide any source in their paper.

2.5 Subjectivity Classification

The *subjectivity* is closely related to SA and opinion mining. According to [Liu, 2012] *subjective* sentence expresses personal feelings, views or beliefs and *objective* sentence holds some factual information about the world. The subjective sentence can be expressed in many ways, e.g., opinions, emotions, stances, allegations, desires, belief, suspicions or speculations [Wiebe et al., 1999, Wiebe, 2000, Riloff et al., 2006]. For example, "*There is one police station in our town.*" is an example of objective sentence and "*Our police are really bad at their job.*" is an example of subjective sentence.

The goal of ***subjectivity classification*** task is to determine whether a sentence is subjective or objective [Wiebe et al., 1999, Wiebe and Riloff, 2005, Liu, 2012, Feldman, 2013, Pang and Lee, 2008, Riloff and Wiebe, 2003]. Nowadays, the subjectivity classification task is considered by some researchers [Medhat et al., 2014] as the first step in sentiment classification to filter out objective sentences that are assumed (incorrectly) to express or imply no opinion.

It is important to note that *subjective* text and *opinionated* text are not equal, although both concepts have a wide intersection. The *opinionated* text expresses or implies positive or negative sentiment. For example, a subjective sentence does not have to necessarily contain any sentiment, as shown in the following sentence "*I think he should visit his doctor*". Similarly, an objective sentence can imply opinion or sentiment thanks to desirable or undesirable facts [Liu, 2012, Feldman, 2013]. For example, "*I did not have to repair my Ford for ten years.*" implies positive sentiment towards the car because of the desirable fact that the car is reliable, the same applied for the following objective sentence "*In XY store I bought a new computer and they gave me five PC games for free.*" also implies positive sentiment. The opinions in the previous objective sentences were *implicit opinions*. Of course it is much easier to detect sentiment in subjective text because it is much more often expressed directly unlike in the case of sentiment in objective text and researchers often do not distinguish between *subjective* and *opinionated* sentences and treat them as equal.

Next, we describe a common dataset for this task and then briefly introduce some approaches from the earliest (which are mentioned in [Liu, 2012]) to the current state-of-the-art solutions. The traditional supervised machine learning is most often applied to the problem of *subjectivity classification.*

2.5.1 Cornell Movie Review Datasets

Here, we describe one well known and commonly used[9] *Cornell Movie Review Dataset*[10] [Pang et al., 2002, Pang and Lee, 2004, 2005]. The corpora consist of two datasets for SA and one dataset for the task of sentence subjectivity classification. The dataset for subjectivity classification consists of 5, 000 subjective and 5, 000 objective sentences. To gather subjective sentences, they downloaded movie reviews sentences or phrases. To obtain objective data, they took sentences from plot summaries of movies from the Internet Movie Database[11].

Similar approach was applied in [Ng et al., 2006] and the analogical approach was also implemented in [Wiebe and Wilson, 2002] and in [Palshikar et al., 2016].

2.5.2 Sentence Subjectivity Classification Approaches

In the very early work, Wiebe et al. [1999] used the Naive Bayes classifier with binary features expressing a presence of a pronoun, an adjective, a cardinal number, a modal other than *will* and an adverb other than *not*. The co-occurrence of words and punctuation marks with subjective and objective sentences.

The semi-unsupervised approach is proposed in [Wiebe, 2000] and it is based on a presence of subjective expressions in a sentence to determine its subjectivity. An initial seed of subjective expression was expanded by searching similar words, which were also likely to be subjective. Riloff et al. [2006] created a rule-based method for subjectivity classification using learned patterns of subjective expressions. In [Barbosa and Feng, 2010], the authors focused on the subjectivity of Tweets. They used traditional features such as word n-grams and part-of-speech tags.

The *Cornell Movie Review Dataset* is usually used for an evaluation of the latest state-of-the-art approaches. To the best of our knowledge, the best results on this dataset were achieved by *AdaSent* model [Zhao et al., 2015] with 95.5% of accuracy. AdaSent stands for a *self-adaptive hierarchical sentence model*, which is a model for a representation of a sentence meaning. AdaSent is inspired by the gated recursive convolutional neural network [Cho et al., 2014b] and forms the representation of the sentence from a traditional word embeddings (e.g., word2vec, GloVe or fastText) but can incorporate the order of words in the sentence and thus improve results on downstream tasks like subjectivity classification.

[9] According to authors, more than 100 papers used their dataset until year 2012. The paper describing the dataset has almost 3, 500 citations according to `https://scholar.google.com` (April 2020).

[10] Available at `http://www.cs.cornell.edu/people/pabo/movie-review-data/`

[11] `https://www.imdb.com`

Cer et al. [2018] recently achieved 93.9% of accuracy with another model for sentence semantic representation, called *Universal Sentence Encoder*. The model produces a fixed length vector (sentence embeddings) for an input sentence.

2.6 Comparative Sentiment Analysis

Sentiment or opinion in a text does not necessarily have to be expressed directly, but a comparison can be used instead. For example, "*Apple iPhone Xs is super reliable*" is a typical *regular opinion* and "*Apple iPhone Xs is much more reliable than Samsung Galaxy S9*" is a typical *comparative opinion* as we defined in Section 2.2.3. Such sentence contains the *comparative opinion*. The goal of this task is to identify sentences that contain comparative opinions, extract the comparative opinions expressed in the sentences and select the preferred entities (Apple iPhone Xs, in our example) [Liu, 2012, 2006, Feldman, 2013].

A comparative opinion expresses a relation of similarities or differences between two or more entities and/or preference of the opinion holder based on some of the shared aspects of the entities. The superlative or comparative form of an adjective or adverb (e.g., *better, more, less, best, cheapest*) are usually used to express the opinion but not always (e.g., *prefer, win*) [Jindal and Liu, 2006a,b, Liu, 2006].

2.6.1 Types of Comparisons

Comparative opinions are created by using several types of comparisons. There are two main types: *gradable comparison* and *non-gradable comparison* [Jindal and Liu, 2006a, Kennedy, 2004, Liu, 2012, 2006].

Gradable comparison expresses relationships where compared entities have some order or rank. This group can be further divided into three subgroups:

1. **Non-equal gradable comparison**: In this group, a set of entities is ranked with a relation of the type *greater* or *less than* a different set of entities. The ranking is based on some of their shared aspects or features. For example, "*Apple iPhone Xs has much more memory than Samsung Galaxy S9*". Also, preferences belong to this group, e.g., "*I prefer Nokia rather than iPhone*".

2. **Equative comparison**: A relation of the type *equal to* is here used for entities. These entities are equal, based on some of their shared aspects of features. For example, "*Apple iPhone Xs and Samsung Galaxy S9 have the same screen size*".

3. **Superlative comparison**: A relation of the type *greater or less than all others* is used to rank one entity over *all other* entities. For example, "*iPhone Xs is the best iPhone model compared with previous models*".

Non-gradable comparison: It represents comparing the relation between two or more entities but does not rank them or assign them any order. This group can also be divided into three sub-groups:

1. **Similarity**: Two or more entities have one aspect or feature which is *similar* or *different*. For example, "*Apple iPhone Xs has a different amount of memory than Samsung Galaxy S9*".

2. **Substitutability**: Two or more entities have different aspects or features, but these aspects are substitutable. For example, "*My iPhone Xs uses a Wi-fi for internet connection, but his Samsung Galaxy S9 can also be connected to internet using 5G network*".

3. **Uniqueness**: Entity has a particular aspect that another entity does not have. For example, "*My iPhone Xs has a front camera, but Samsung Galaxy S9 does not have it*".

In [Liu, 2012, 2006] are the types of comparisons discussed and described much more deeply.

2.6.2 Task Objective

The objective of the *comparative sentiment analysis* [Liu, 2012, Liu et al., 2010, Jindal and Liu, 2006b] is for a given opinion document d, discover in d all comparative opinion sextuples of the form:

$$(E_1, E_2, A, PE, h, t)$$

where E_1 and E_2 are the entity sets that are compared based on their shared aspects A. $PE \in \{E_1, E_2\}$ is the preferred entity set of the opinion holder h and t is a time when the opinion is expressed. In the case of *superlative comparison*, where one entity set is implicit (it is not present in the text) special set U is used to denote it. For *equative comparison*, a special symbol $EQUAL$ is used as a value of the PE member. For example, using the following sentence:

Author: Nick, 25/10/2019

"*iPhone has much more memory than Samsung S9 and Honor 20 Pro.*"

the desired output should look like this:

$$(E_1, E_2, A, PE, h, t) =$$

(iPhone, {Samsung S9, Honor 20 Pro}, memory, iPhone, Nick, 25/10/2019)

The entity set E_1 contains only *iPhone*, the entity set E_2 consists of *Samsung S9* and *Honor 20 Pro*. The shared aspect set A, which is compared, contains *memory*. The set of proffered entities PE contains *iPhone*. The opinion holder h is *Nick* and time t is *25/10/2019*.

Comparative Sentence Identification

The problem of *comparative sentence identification* may seem trivial because almost all comparative sentences contain comparative or superlative adjective or adverbs keywords (e.g., *better, bigger, best*) as is shown in [Jindal and Liu, 2006a] but there two types of sentences which make the detection based only on keywords difficult. The first one is that there are many sentences which contain comparative keywords, but they are not comparative sentences, e.g., "*The words less and fewer both mean the same thing*" and the second one is that some comparative sentences do not contain any comparative keyword, e.g., "*In KFC I have to buy french fries, but in McDonald's restaurant they are included in every menu*".

Jindal and Liu [2006a] were able to detect comparative sentences on their dataset using only a list of keywords with high recall but with low precision. Further, in [Jindal and Liu, 2006b] they classify the *comparative sentences* into four classes, i.e., *non-equal gradable, equative, superlative* and *non-gradable*, see Section 2.6.1. They achieved an 80% of F_1 score.

Other related work to this problem can be found in [Li et al., 2010b, Yang and Ko, 2011, Gupta et al., 2017].

Preferred Entities Identification

Since the comparative sentences in most cases compare sets of entities, it is not much meaningful to assign one sentiment label to the whole sentence because they usually do not express a direct positive or negative opinion. So the goal in the *preferred entities identification* task is to rank the sets according to their shared aspects, which are compared. In most comparative sentences, there are two entity sets and the ranking means to find the preferred entity set. For some real life application the positive sentiment label may be assigned to aspects of the preferred entity set and negative to not preferred aspects [Liu, 2012, 2006].

Liu [2012, 2006] describes an approach from [Ding et al., 2009, Ganapathibhotla and Liu, 2008] for identification of the preferred entity set, which is an extension of lexicon-based approach for *aspect-based sentiment analysis* of regular opinions. The described approach is modified to be used with comparative sentences.

2.7 Other Tasks

In this section, we briefly describe some other tasks related to SA.

2.7.1 Lexicon Generation

Words that carry information about sentiment are crucial for SA and the other related tasks. These words are called *sentiment words* or *opinion words*. For example, *excellent, nice, beautiful* and *cool* are positive sentiment words and *awful, bad* and *nasty* are negative sentiment words. Along with sentiment words, there are some specific phrases and idioms which also hold sentiment, for example, *"He was on cloud nine"* means that someone is happy, which implies positive sentiment. A list of such words, along with their sentiment orientation, is called *sentiment lexicon*. Sentiment lexicon can be directly used for solving one of the SA tasks (see Chapter 5) or as a source for feature extraction for supervised learning algorithms.

Sentiment lexicon consists of a set of tuples of a *lexical unit* (word, phrase or idiom) and its *sentiment score* [Liu, 2012, Singh et al., 2018, Medhat et al., 2014]. The sentiment score can be represented in the following ways:

1. A binary indication to positive or negative sentiment polarity, for example: excellent = 1, nice = 1, bad = 0.

2. A set of predefined values, like *very negative, negative, neutral, positive* and *very positive*, for example: excellent = *very positive*, nice = *positive*, bad = *negative*.

3. A real number from a predefined interval, for example, $[-1, 1]$ where negative values refer to negative sentiment and positive values refers to positive sentiment, for example: excellent = 0.78, bad = -0.35.

There are generally three main approaches for generating or obtaining sentiment lexicon [Liu, 2012, Medhat et al., 2014]. The *manual approach* is very time-consuming and expensive, thus it is often used as a verification for the other two automatic approaches, i.e., *dictionary-based approach* and *corpus-based approach*.

Dictionary-Based Approach

The dictionary-based approach uses a small set of manually selected sentiment words with their sentiment orientation or score. This set is called a *seed*.

A typical algorithm which uses this approach expands the initial set by searching for synonyms and antonyms from some dictionary, e.g., WordNet [Miller et al., 1990, Miller, 1995] or another prepared source [Mohammad et al., 2009]. The newly found words are then added to the initial set. This process is iteratively repeated with the newly expanded sentiment lexicon. The process is stopped when the required amount of words is obtained or when no new words are found. Some manual control is usually performed to check the generated lexicon and correct errors.

The main advantage of this approach is that a large list of sentiment words can be easily obtained. The disadvantage, on the other hand, is that such a list contains errors that have to be manually fixed. The major disadvantage is that the list obtained with this approach contains only domain and context-independent sentiment words, which can be problematic for some applications [Liu, 2012]. For example, in the sentence "The apartment is quite small", the word *small* can imply negative sentiment and in the sentence "The USB flash drive is really small" the word *small* means positive feature.

Examples of this approach and its modification can be found in [Hu and Liu, 2004, Valitutti et al., 2004, Kim and Hovy, 2004, Mohammad et al., 2009, Blair-Goldensohn et al., 2008, Rao and Ravichandran, 2009] the summary of these papers is in [Liu, 2012].

Corpus-Based Approach

The corpus-based approaches are trying to solve or help with the problem of the domain and context specific sentiment words. The idea of this approach is to find some syntactic patterns or other patterns that occur together along with already known sentiment words in a given large corpus. There are usually two main scenarios in which the corpus-based approach is used [Liu, 2012, Medhat et al., 2014]:

1. Given a seed list of known (often domain and context independent) sentiment words and a domain corpus. The other domain and context dependent sentiment words should be discovered from the domain corpus.

2. Given a sentiment lexicon (with domain and context independent) sentiment words and a domain corpus. The task is to adapt the given sentiment lexicon to a new one using the given domain corpus.

The corpus-based approach is more suitable for creating domain specific sentiment lexicons or utilizing existing context and domain independent lexicons in order to find domain specific sentiment words. The reason for this claim is that is quite difficult to prepare very large and diverse corpus which covers all English words but in a such a case corpus-based approach can be used [Liu, 2012, Medhat et al., 2014].

Available Lexicons

Thanks to the previous research, there is several of sentiment and emotion lexicon and most of them are publicly available. In this section, we provide a brief overview of these lexicons. Sentiment lexicons for English are the most common, for example: General Inquirer lexicon [Stone et al., 1966], MPQA [Wilson et al., 2005], Bing Liu [Hu and Liu, 2004], AFINN [Nielsen, 2011], Sentiment140 [Kiritchenko et al., 2014], NRC Hashtag Sentiment [Kiritchenko et al., 2014], SentiWordNet [Baccianella et al., 2010] or SentiStrength [Thelwall et al., 2012]. For other languages some lexicons exist as well, for example, for Spanish: ML-SentiCon [Cruz et al., 2014] or iSOL [Molina-González et al., 2013].

Another type of lexicon is a lexicon with *valence arousal* and *dominance* values, which is based on Valence-Arousal-Dominance (VAD) emotion model, see Section 2.4.2. There are English VAD lexicons [Mohammad, 2018, Bradley and Lang, 1999, Warriner et al., 2013] but also Spanish [Redondo et al., 2007, Guasch et al., 2016], Dutch [Moors et al., 2013, Brysbaert et al., 2014], German [Vo et al., 2009, Schmidtke et al., 2014], French [Gilet et al., 2012, Monnier and Syssau, 2014], Finnish [Söderholm et al., 2013], Polish [Riegel et al., 2015], Chinese [Yu et al., 2016] and Portuguese [Soares et al., 2012].

Lastly the emotion lexicons exist as well for English; NRC Word-Emotion Association Lexicon [Mohammad and Turney, 2013], NRC-10 Expanded [Bravo-Marquez et al., 2016], NRC Hashtag Emotion Association Lexicon [Mohammad and Kiritchenko, 2015] or Emotion lexicon [Mohammad and Turney, 2010]. These lexicons contain score for each of predefined emotions.

2.7.2 Opinion Spam Detection

Along with the rise of social networks and the internet itself, harmful activities also appeared. Opinions, reviews and comments can affect real life in a good or even in a bad way. Positive reviews about a certain product can boost its sales and negative reviews can stop its sales. Due to the anonymity of the internet, artificial opinions and comments can be abused, to promote or discredit some target products, services organizations and individuals. Such behavior can be dangerous for individuals, companies or even countries. People posting *fake opinions* or *reviews* are called *opinion spammers* and their activities are called *opinion spamming* [Jindal and Liu, 2007, 2008, Liu, 2012]. The goal of this task is to detect such spam opinions. Another related work and detail description of this task can be found in [Liu, 2012, Rayana and Akoglu, 2015, Ren and Ji, 2017, Crawford et al., 2015].

2.7.3 Sarcasm Detection

Recognize the real meaning of some expressions in a natural language like irony, sarcasm or satire is a challenging task not only for computers but sometimes even for humans. These terms are closely related and (sometimes are considered as interchangeable) we do not distinguish between them [Reyes et al., 2012]. The goal of *Sarcasm Detection* is to identify sarcastic sentences or other pieces of text [Davidov et al., 2010, Reyes et al., 2012, Liu et al., 2010]. In the context of SA, the meaning of a positive expression is usually intended to be negative and vice versa [Liu, 2012] this is the main reason why the task is so challenging and difficult.

Other related work can be found in [Zhang et al., 2018, Van Hee et al., 2018] and for Czech in [Ptáček et al., 2014].

2.7.4 Opinion Summarization

The next step after one of the polarity detection tasks is a summarization of obtained opinions. In most cases, in practice, the company or user will also want to know the overall opinion, i.e., what is the distribution of sentiments towards a specific entity and its aspects. Such summarization can be composed of two components – *qualitative* component which covers expressed sentiments towards a structured group of entities or aspects and *quantitative* component which gives information in numbers about expressed sentiments [Liu, 2012].

For example, a summary for some camera reviews could be that 1563 reviews expressed a positive sentiment, 132 expressed negative sentiment and 15 expressed neutral sentiment towards *GENERAL* aspect of entity *Sony Camera x5*. Other aspects of the camera can be summarized, as is shown in Table 2.7. For a detailed description of opinion summarization see Section 2.2 and Chapter 7 in [Liu, 2012].

Aspect	General			Weight			Memory Size		
Sentiment	pos	neg	neu	pos	neg	neu	pos	neg	neu
Count	1563	132	15	642	967	351	1348	692	118

Table 2.7: Summary for *Sony Camera X5*.

Chapter 3

Machine Learning for Classification

Machine learning is an essential part of NLP, including all SA tasks. In this chapter, we will describe some algorithms and methods of machine learning used in SA and text classification. We focus on *supervised machine learning* since it dominates in SA.

3.1 Naive Bayes Classifier

Naive Bayes (NB) classifier is a simple supervised machine learning algorithm based on Bayes' theorem that allows to estimate conditional probability of some event, it can be written as:

$$p(A \mid B) = \frac{p(B \mid A)p(A)}{p(B)} \tag{3.1}$$

where A and B are events, $p(A)$ (*prior* probability) and $p(B)$ (*evidence*) are marginal probabilities, $p(A \mid B)$ is a conditional probability of A given B also called the *posterior* probability and $p(B \mid A)$ is a probability of B given A also called *likelihood*.

The NB classifier assumes that features are independent of each other, which is often an invalid assumption in many applications where NB classifier is used, especially in the NLP field. Despite this often incorrect assumption, NB classifier can achieve quite good results and in the early research, it was quite a popular choice for SA. For NLP, we can rewrite the Bayes' theorem given by equation 3.1 as:

$$p(c \mid d) = \frac{p(d \mid c)p(c)}{p(d)} \tag{3.2}$$

where $c \in \mathbf{C}$ is a class label and d is a document. $\mathbf{C}$ is a finite set of possible class labels. The Naive Bayes classifier assigns a given document d into class

$\hat{c}$ that fulfills the following expression:

$$\begin{aligned}\hat{c} = \arg\max_{c \in \mathbf{C}} p(c \mid d) &= \arg\max_{c \in \mathbf{C}} \frac{p(d \mid c)p(c)}{p(d)} \\ &= \arg\max_{c \in \mathbf{C}} p(d \mid c)p(c) \\ &= \arg\max_{c \in \mathbf{C}} p(x_1, x_2, \ldots, x_n \mid c)p(c)\end{aligned} \tag{3.3}$$

where $x_1, x_2, \ldots, x_n$ are features representing the document d. Because the $p(d)$ is a constant, we can drop the denominator. We assume that the features are independent of each other and thus we can apply the *conditional independence* rule and rewrite the probability $p(x_1, x_2, \ldots, x_n \mid c)$ as follows:

$$p(x_1, x_2, \ldots, x_n \mid c) = p(x_1 \mid c) \times p(x_2 \mid c) \times \cdots \times p(x_n \mid c) \tag{3.4}$$

The Naive Bayes classifier then can classify document d into class $\hat{c}$ as follows:

$$\hat{c} = \arg\max_{c \in \mathbf{C}} p(c) \prod_{i=1}^{|\mathbf{X}|} p(x_i \mid c) \tag{3.5}$$

where $p(c)$ is the prior probability of class c, $x_i \in \mathbf{X}$ and $\mathbf{X}$ is a set of features for the document d.

3.2 Support Vector Machines

Support Vector Machines (SVM) [Cortes and Vapnik, 1995] is another supervised machine learning algorithm. The original paper was introduced only for binary classification, but it can be easily extended for any number of classes using the *one-vs-all* technique. When the one-vs-all technique is used, k (number of classes) binary classifiers are trained, i.e., one for each class. Here we describe only Linear SVM that assumes that the data are linearly separable, but the non-linear version exists too, see [Cortes and Vapnik, 1995].

The basic idea of training SVM is to find an optimal hyperplane, represented by a normal vector $\mathbf{w}$, that separates documents (training data represented as vectors) in one class from documents in the other class. The important aspect of the separating vector $\mathbf{w}$ is that SVM tries to find it in a way that the separation *margin* between the vectors of documents in different classes is as large as possible, as shown in Figure 3.1, hence SVM is also known as *maximum margin classifier*.

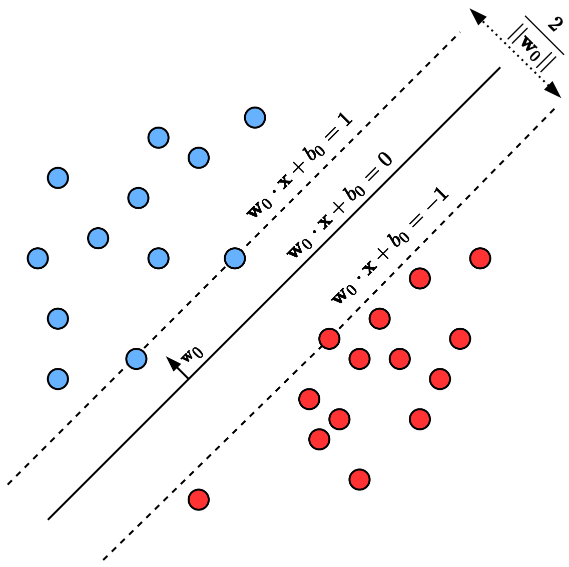

$\mathbf{w_0} \cdot \mathbf{x} + b_0 = 1$
$\mathbf{w_0} \cdot \mathbf{x} + b_0 = 0$
$\mathbf{w_0} \cdot \mathbf{x} + b_0 = -1$
$\mathbf{w_0}$
$\frac{2}{||\mathbf{w_0}||}$

Figure 3.1: Support Vector Machines optimal hyperplane principle illustration. The optimal hyperplane is given by the normal vector $\mathbf{x_0}$ and scalar b_0.

Any hyperplane can be defined as a set of vectors $\mathbf{x}$ (points) that satisfy the following expression:

$$\mathbf{w} \cdot \mathbf{x} + b = 0 \tag{3.6}$$

where $\mathbf{w}$ is the normal vector to the hyperplane and the $\frac{b}{||\mathbf{w}||}$ determines the offset of the hyperplane from the origin along the normal vector $\mathbf{w}$.

First, let us introduce a notation we follow for a training dataset that consists of the following training examples $\{(\mathbf{x}^{(1)}, y^{(1)}), \ldots, (\mathbf{x}^{(m)}, y^{(m)})\}$ where $y^{(i)} \in \{-1, 1\}$ (binary classification), m is a size of the training dataset and $\mathbf{x}^{(j)}$ is a vector of features $[x_1, x_2, \ldots, x_n]$, thus a particular feature i of training example $\mathbf{x}^{(j)}$ is referred[1] as $\mathbf{x}_i^{(j)}$. Given a linearly separable training dataset $\{(\mathbf{x}^{(1)}, y^{(1)}), \ldots, (\mathbf{x}^{(m)}, y^{(m)})\}$ then there must exist the normal vector $\mathbf{w}$ and scalar b that satisfy the following inequalities:

$$\begin{array}{rclll} \mathbf{w} \cdot \mathbf{x}^{(i)} + b & \geq & 1 & \text{if} \;\; y^{(i)} = & 1 \\ \mathbf{w} \cdot \mathbf{x}^{(i)} + b & \leq & -1 & \text{if} \;\; y^{(i)} = & -1 \end{array} \tag{3.7}$$

where $\mathbf{w} \cdot \mathbf{x}^{(i)} + b = 1$ is a boundary, i.e., *support vector* and all training examples $\mathbf{x}^{(i)}$, for which $y^{(i)} = 1$, lies either on or above the boundary,

[1]We use this notation also in the next sections.

$\mathbf{w} \cdot \mathbf{x}^{(i)} + b = -1$ is a boundary, i.e., *support vector* and all training examples $\mathbf{x}^{(i)}$, for which $y^{(i)} = -1$, lies either on or below the boundary. The equation 3.7 can be rewritten as follows:

$$y^{(i)}(\mathbf{w} \cdot \mathbf{x}^{(i)} + b) \quad \geq \quad 1, \qquad i = 1, \ldots, n \tag{3.8}$$

And finally, the desired optimal hyperplane (defined by a vector $\mathbf{w}_0$ and scalar b_0) is the unique one hyperplane in the way that it separates the training data with a maximal margin and it is given by:

$$\mathbf{w}_0 \cdot \mathbf{x} + b_0 = 0 \tag{3.9}$$

The distance $\rho(\mathbf{w}, b)$ between the projections of the training vectors of two different classes is given by:

$$\rho(\mathbf{w}, b) = \min_{x;y=1} \frac{\mathbf{x} \cdot \mathbf{w}}{||\mathbf{w}||} - \max_{x;y=-1} \frac{\mathbf{x} \cdot \mathbf{w}}{||\mathbf{w}||} \tag{3.10}$$

The parameters that are being searched are the $\mathbf{w}_0$ and b_0. The distance between the two support vectors from equation 3.7 for the optimal hyper plane is given by:

$$\rho(\mathbf{w}_0, b_0) = \frac{2}{||\mathbf{w_0}||} \tag{3.11}$$

In order to maximize the distance $\rho(\mathbf{w}, b)$ (obtain $\mathbf{w}_0$ and b_0) we want to minimize $||\mathbf{w}||$. After getting the optimal hyperplane defined by $\mathbf{w}_0$ and b_0 classification of example $\mathbf{x}^{(k)}$ into class $y \in \{1, -1\}$ is defined by the following function:

$$f(\mathbf{x}^{(k)}) = \operatorname{sgn}(\mathbf{w_0} \cdot \mathbf{x}^{(k)} + b_0) \tag{3.12}$$

3.3 Logistic Regression

Logistic regression is a well known supervised classification algorithm (despite the word *regression* in its name). The basic logistic regression can be used for a binary classification, the *multinomial logistic regression* allows classification into more than two classes. Here, we describe the basic version for two classes. We describe this method in relative detail because most of its basic principles and components are applied in more complex algorithms, concretely in neural networks, see Section 3.4. Along with logistic regression, we also explain general terms and concepts like *cost function, gradient descent* and *regularization* that are common in machine learning in general.

3.3.1 Generative and Discriminative Classifiers

Logistic regression is a type of classifier referred to as a *discriminative* classifier, the classifiers of the second type are called *generative* classifiers[2]. The generative classifiers model (*"generate"*) the distribution of individual classes. On the other hand, discriminative algorithms learn a decision boundary between the classes, i.e., it only learns how to distinguish between the classes based on their features.

More formally, both classifiers predict the conditional probability $p(c \mid d)$ of class c given the input document d (technically by features representing the document), but both of them computes the probability differently. Generative models learn to model the joint probability distribution $p(d, c)$ and compute the conditional probability $p(c \mid d)$ to predict the class c. Such example is the Naive Bayes classifier given by equation 3.5. The discriminative classifiers directly model the conditional probability $p(c \mid d)$ [Ng and Jordan, 2002, Jurafsky and Martin, 2009].

3.3.2 Logistic Regression Model

Generally, logistic regression predicts (estimates) the most likely class for input vector of features $\mathbf{x}^{(i)}$ representing the input document d_i by computing the probability $p(y \mid \mathbf{x}^{(i)})$. To recall, we use the same[3] notation for training examples as in Section 3.2 with the description of SVM classifier, i.e., $\{(\mathbf{x}^{(1)}, y^{(1)}), \dots, (\mathbf{x}^{(m)}, y^{(m)})\}$ denotes the training examples, $y^{(i)} \in \{0, 1\}$, m is a size of the training dataset and $\mathbf{x}^{(j)} \in \mathbb{R}^n$ is a vector of features $[x_1, x_2, \dots, x_n]$ thus a particular feature i of training example $\mathbf{x}^{(j)}$ is referred to as $\mathbf{x}_i^{(j)}$. All vectors of training examples can also be written as the matrix $\mathbf{X} \in \mathbb{R}^{n \times m}$ and all labels as a vector $\mathbf{y} \in \mathbb{R}^m$.

The logistic regression model optimizes weights $\mathbf{w}$ and a bias parameter b. The model is learning by minimizing an error on training examples. The error is computed by the objective function, e.g., *cross-entropy loss* function. The algorithm that optimizes the parameters according to the objective function is, for example, *stochastic gradient descent* or *gradient descent* [Jurafsky and Martin, 2009].

For input vector $\mathbf{x}$ we want to compute the probability $p(y = 1 \mid \mathbf{x})$, i.e., the input $\mathbf{x}$ belongs to class $y = 1$ which can represent, for example, that the input document is *spam* and for $y = 0$ that the input document is *non-spam*. To be exact, the model estimates the probability $\hat{p}(y = 1 \mid \mathbf{x})$ of the true probability $p(y = 1 \mid \mathbf{x})$. The prediction of the classifier on a test example is computed in two steps. First, the $z \in \mathbb{R}$ scalar term is computed

[2]Here, for the explanation, we will consider document classification, i.e., the input for any classifiers is a document d and output is its class c.

[3]The only difference is that $y^{(i)} \in \{0, 1\}$ instead of $y^{(i)} \in \{-1, 1\}$.

from the input $\mathbf{x}$ and model's parameters $\mathbf{w}$ and b in the following way:

$$z = \mathbf{w}^\mathsf{T}\mathbf{x} + b \tag{3.13}$$

Since the z is a real number, we need to convert it into a probability output $\hat{y}$. The $\hat{y}$ represents the estimation of the true y. The estimation is achieved by applying the *sigmoid* function as follows:

$$\hat{p}(y = 1 \mid \mathbf{x}) = \hat{y} = \sigma(z) = \frac{1}{1 + e^{-z}} \tag{3.14}$$

The sigmoid function maps the input real number to the range $[0, 1]$, see Figure 3.2. The sigmoid function is also called the *logistic function*, hence the name *logistic regression*.

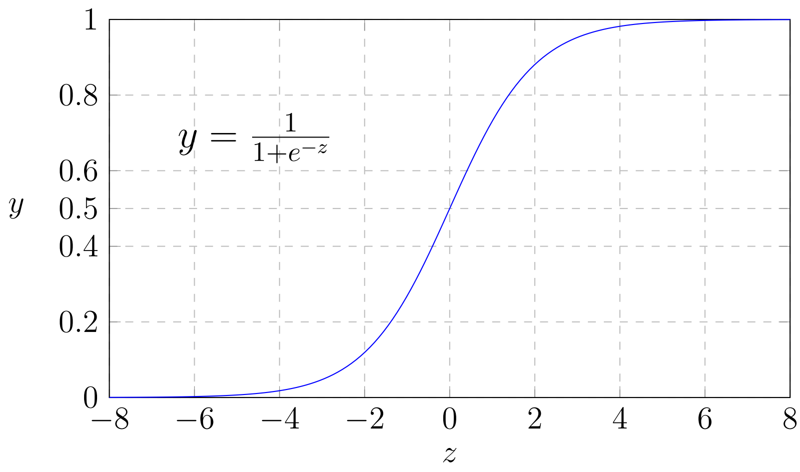


Figure 3.2: Sigmoid function illustration.

As we mentioned, the output of the model is the estimated probability $\hat{p}(y = 1 \mid \mathbf{x})$ of the input $\mathbf{x}$ being assigned to the class $y = 1$ thus for the binary classification the probability $\hat{p}(y = 0 \mid \mathbf{x})$ can be computed as follows

$$\hat{p}(y = 0 \mid \mathbf{x}) = 1 - \hat{p}(y = 1 \mid \mathbf{x}) \tag{3.15}$$

and $\hat{p}(y = 1 \mid \mathbf{x})$ plus $\hat{p}(y = 0 \mid \mathbf{x})$ sum up to one:

$$\hat{p}(y = 1 \mid \mathbf{x}) + \hat{p}(y = 0 \mid \mathbf{x}) = 1 \tag{3.16}$$

Finally, the predicted class y is 1 if the estimated probability $\hat{p}(y = 1 \mid \mathbf{x})$ is greater than threshold 0.5 (which is called the *decision boundary*), 0 otherwise, see Figure 3.3, it can also be written as follows:

$$\text{prediction} = \begin{cases} 1 & \text{if } \hat{p}(y = 1 \mid \mathbf{x}) > 0.5 \\ 0 & \text{otherwise} \end{cases} \tag{3.17}$$

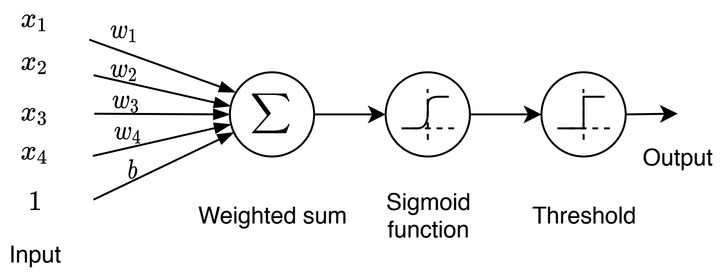


Figure 3.3: Logistic regression model.

3.3.3 Cost Function

In the previous section, we assumed that the parameters $\mathbf{w}$ and b are already optimized. Logistic regression is a supervised machine learning algorithm, so in order to learn its parameters, two components are needed, i.e., the *cost function* and the *optimization algorithm*. The *cost* or *loss* function is a metric that tells us how different the outputs are for the model's training data compared to the true (gold) labels. Logistic regression uses *cross-entropy* cost function. The second component is the optimization algorithm that updates the model's parameters to minimize the cost function. Usually, the *gradient descent* or *stochastic gradient descent* is used as the optimization algorithm [Jurafsky and Martin, 2009].

Given one training example $\mathbf{x}$, the gold label y and the prediction of the model $\hat{y}$ (i.e., the $\hat{p}(y = 1 \mid \mathbf{x})$ probability), the *cross-entropy* loss function $\mathcal{L}_C(\hat{y}, y)$ is defined as follows:

$$\mathcal{L}_C(\hat{y}, y) = -(1-y)\log(1-\hat{y}) - y\log\hat{y} \tag{3.18}$$

Using equations 3.14 and 3.13 we can rewrite the cross-entropy loss function as follows:

$$\mathcal{L}_C(y, \mathbf{x}; \mathbf{w}, b) = -(1-y)\log(1-\sigma(\mathbf{w}^\mathsf{T}\mathbf{x}+b)) - y\log\sigma(\mathbf{w}^\mathsf{T}\mathbf{x}+b) \tag{3.19}$$

The goal of the optimization algorithm, e.g., gradient descent, is to optimize the parameters $\mathbf{w}$ and b. The previous definition of cross-entropy loss (equations 3.18 and 3.19) is focused only on one training example. The overall cost function $J(\Theta)$, for the entire training dataset, that we want to minimize (and thus find the optimal parameters) is defined as the average cross-entropy over all training examples:

$$J(\Theta) = \frac{1}{m}\sum_{i=1}^{m} \mathcal{L}_C(y^{(i)}, \mathbf{x}^{(i)}; \Theta) \tag{3.20}$$

where $\Theta = (\theta_1, \theta_2, \dots, \theta_n, \theta_{n+1})$ is a vector that represents the model parameters $\mathbf{w}$ and the bias term b that are being optimized. The dimension of $\mathbf{w}$ is n.

3.3.4 Learning and Gradient Descent

The learning (optimization of the parameters) is then done by the gradient descent algorithm. Gradient descent computes the gradient of the cost function by computing partial derivative $\frac{\partial J(\Theta)}{\partial \theta_i}$ with respect to each parameter θ_i. Each parameter θ_i is updated in the following way:

$$\theta_i^{t+1} = \theta_i^t - \alpha \frac{\partial J(\Theta)}{\partial \theta_i} \tag{3.21}$$

where θ_i^t is the current parameter value (i.e., before the update), θ_i^{t+1} is the new parameter value after the update, α is a learning rate and $\frac{\partial J(\Theta)}{\partial \theta_i}$ is the partial derivative of the cost function with respect to the parameter θ_i.

The computed gradient is represented by a vector $\nabla_\Theta J(\Theta)$ where each element corresponds to the element in the vector of parameters Θ and contains the partial derivative with respect to that parameter. Using the cost function from equation 3.20, the gradient vector $\nabla_\Theta J(\Theta)$ is computed as the average of partial derivatives for each training example:

$$\nabla_\Theta J(\Theta) = \frac{1}{m} \sum_{i=1}^{m} \nabla_\Theta \mathcal{L}_C(y^{(i)}, \mathbf{x}^{(i)}; \Theta) \tag{3.22}$$

Then, one step (update) of the entire model can be rewritten as follows:

$$\Theta^{t+1} = \Theta^t - \alpha \nabla_\Theta J(\Theta) \tag{3.23}$$

Gradient descent is the iterative algorithm that can be stopped when the gradients are less than some predefined value ϵ or when the cost function does not change by a predefined value over the iterations or when the cost function starts to grow on some held-out data [Jurafsky and Martin, 2009].

The cross-entropy loss function for logistic regression is convex. Thanks to this property, it is guaranteed that the gradient descent algorithm finds the (global) minimum.

The described basic version of gradient descent is computationally expensive since to make one iteration (update), the gradient needs to be computed for all training examples. *Stochastic gradient descent* is another version of gradient descent that performs the update of the parameters for each training example $\mathbf{x}^{(i)}$. The gradient $\nabla_\Theta J(\Theta)$ for one training example $\mathbf{x}^{(i)}$ is then computed (technically estimated) as follows:

$$\nabla_\Theta J(\Theta) = \nabla_\Theta \mathcal{L}_C(y^{(i)}, \mathbf{x}^{(i)}; \Theta) \tag{3.24}$$

Another property of stochastic gradient descent is that, to a certain extent, it can be used even for non-convex loss functions (e.g., neural networks) since it can get out of some local optima of the function.

Alternatively, *mini-batch gradient descent* can be applied as well. Instead of computing gradient for only one training example or all training examples, mini-batch gradient descent computes gradient for a batch of l training examples and updates the model's parameters after each batch.

$$\nabla_\Theta J(\Theta) = \frac{1}{l}\sum_{i=1}^{l} \nabla_\Theta \mathcal{L}_C(y^{(i)}, \mathbf{x}^{(i)}; \Theta) \tag{3.25}$$

3.3.5 Regularization

In order to prevent the model from *overfitting*, a regularization technique is often used. It allows the model to *generalize* on unseen test data [Jurafsky and Martin, 2009]. The typical way of implementing the regularization is by adding a new regularization term $\Sigma(\Theta)$ to the cost function. The cost function is then given by:

$$J(\Theta) = \frac{1}{m}\sum_{i=1}^{m}(\mathcal{L}_C(y^{(i)}, \mathbf{x}^{(i)}; \Theta) + \lambda\Sigma(\Theta)) \tag{3.26}$$

where λ is a hyper-parameter that controls the strength of the regularization. If $\lambda = 0$ we get the original cost function without any regularization. There are two standard methods for computing the regularization term $\Sigma(\Theta)$. (1) ℓ_1 *regularization* is the sum of absolute values of the parameters and it is given by:

$$\Sigma(\Theta) = ||\Theta||_1 = \sum_{i=1}^{n+1} |\theta_i| \tag{3.27}$$

(2) The ℓ_2 *reguralization* is computed as follows:

$$\Sigma(\Theta) = ||\Theta||_2^2 = \sum_{i=1}^{n+1} \theta_i^2 \tag{3.28}$$

3.3.6 Multinomial Logistic Regression

Until now, we described logistic regression for binary classification. For multi-class classification, the *multinomial logistic regression* learns a separate set of parameters $\theta_k \in \Theta$ for each class $c_k \in C$, the number of classes is l, i.e., $|C| = l$. Again, the goal is to estimate probability $\hat{p}(y = c \mid \mathbf{x})$ of the true probability $p(y = c \mid \mathbf{x})$ that the input $\mathbf{x}$ belongs to the class c. First, vector $\mathbf{z} = [z_1, z_2, \ldots z_l]$ is computed, where each component of the vector is computed[4] from a set of parameters θ_k for a class c_k and input $\mathbf{x}$ using

[4]In practice, all these operations are vectorized and parameters are in matrices.

the equation 3.13. Next, the vector $\mathbf{z}$ is passed through the *softmax* function that produces the estimated probabilities $\hat{y} = \hat{p}(y = c \mid \mathbf{x}; \Theta)$ for each class. The estimated probability $\hat{p}(y = c \mid \mathbf{x}; \Theta)$ for a specific class c_k is computed as follows:

$$\operatorname{softmax}(z_k) = \hat{p}(y = c_k \mid \mathbf{x}; \Theta) = \frac{e^{z_k}}{\sum_{j=1}^{l} e^{z_j}} = \frac{e^{\Theta_k^\intercal \mathbf{x} + b_k}}{\sum_{j=1}^{l} e^{\Theta_j^\intercal \mathbf{x} + b_j}} \tag{3.29}$$

After applying the softmax function on the vector $\mathbf{z}$, the output is a vector of probabilities for the input $\mathbf{x}$ assigned to the corresponding classes. The cross-entropy loss function $\mathcal{L}_{CRE}(\hat{y}, y, \mathbf{x})$ for one training example $\mathbf{x}$ is given by:

$$\mathcal{L}_{CRE}(\hat{y}, y, \mathbf{x}) = -\sum_{k=1}^{l} 1\{y = k\} \log \hat{p}(y = k \mid \mathbf{x}; \Theta) \tag{3.30}$$

where $1\{y = k\}$ is equal to 1 if $y = k$, zero otherwise. In other words it is equal to 1 if the input $\mathbf{x}$ is labeled with the gold class c_k.

3.4 Neural Networks

In this section, we describe *neural networks* and their underlying concepts since, nowadays, neural networks became the fundamental machine learning tool for natural language processing. The *neural* in the name originates from the first proposal of an artificial neuron, called *McCulloch-Pitts neuron* [McCulloch and Pitts, 1943]. The McCulloch-Pitts neuron was based on a simplification of the biological neuron.

Generally, neural networks are built from individual units (called neurons) and stacked into layers, together the layers form the entire neural network. Logistic regression and neural network are closely related since a neural network can be seen as a composition of multiple logistic regression models (or other functions) stacked on top of each other, where the units are the individual logistic regression classifiers (taking only the sigmoid output, not 0 or 1 output after applying the threshold). From the other point of view, logistic regression can be considered as a simple neural network [Jurafsky and Martin, 2009]. In the following sections, we discuss different types of neural network architectures, i.e., *feed-forward neural network* in Section 3.4.2, *recurrent neural network* in Section 3.5 and *transformer* architecture in Section 3.7.

3.4.1 Deep Learning

In recent years, a very popular term *deep learning* has emerged in the context of AI and neural networks. It refers to neural networks with many layers (regardless of the type of layer) in neural network architectures, hence *deep learning*. The important idea behind deep learning is that the model is able to learn representations of data, i.e., features are extracted by the network itself (automatically) without any explicit or manual feature engineering. For example, neural networks for image recognition or computer vision contain dozens of layers, where the lower layers can identify simpler features (e.g., edges) and the higher layers can identify more complex features (e.g., parts or even entire objects like digits, letters or faces) from outputs of the lower layers. The deep neural networks are built by composing individual layers that are usually implemented by the *feed-forward neural network*, *recurrent neural network* or *convolutional neural network*, but any type of neural network can be incorporated in general. Deep leaning turned out to be a powerful machine learning technique capable of producing state-of-the-art results in NLP and also in other applications (e.g., computer vision or speech recognition) [Jurafsky and Martin, 2009, Goodfellow et al., 2016].

In the case of NLP, word embeddings are also features that are extracted automatically (usually by a neural network). Similarly to the image processing, different layers in a neural network for NLP can capture different language information. For example, lower layers in the ELMo model (see Section 4.2.1) are better at capturing syntax information and the higher layer is better at capturing context and semantic information.

3.4.2 Feed-Forward Neural Network

Next, we describe an architecture called *feed-forward neural network*, also known as *multilayer perceptron* (MLP). It is composed from individual layers and each layer is built of individual units (neurons). Feed-forward network consists of one *input layer*, one *output layer* and one or more *hidden layers*, see Figure 3.4. Each neuron from one layer is connected with all neurons in the consecutive layer, hence, this network architecture is sometimes called *fully-connected*. These connections are called *weights* and they are parameters of the entire network. As with logistic regression, the goal is to optimize these weights so that the model will produce the desired output.

The MLP with one hidden layer (see Figure 3.4) takes as input vector $\mathbf{x}$ of n features and it is passed through the entire network, i.e., through each neuron. It can be written as follows:

$$\begin{aligned}\mathbf{h} &= \sigma(\mathbf{W_1}\mathbf{x} + \mathbf{b_1}) \\ \hat{\mathbf{y}} &= \text{softmax}(\mathbf{W}_2\mathbf{h} + \mathbf{b_2})\end{aligned} \tag{3.31}$$

where $\mathbf{W}_1$ and $\mathbf{W}_2$ are weight matrices, $\mathbf{b}_1$ and $\mathbf{b}_2$ are bias vectors, $\mathbf{h}$ is an output vector of the hidden layer, $\sigma(\cdot)$ is an activation function and $\hat{\mathbf{y}}$ is a vector of a probability distribution over possible output classes.

First, for each neuron, the weighted sum (scalar) is computed, then the weighted sum is passed through a non-linear function $\sigma(\cdot)$ that is called the

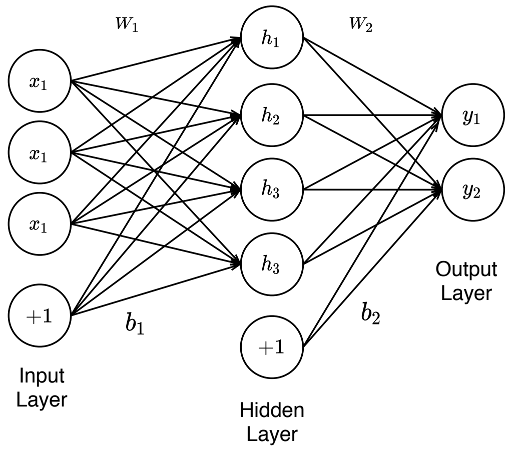


Figure 3.4: Example of feed-forward neural network.

activation function. The example of the activation function is the sigmoid function (used in logistic regression) but other functions like ReLU (Rectified Linear Unit), see equation 3.33, or hyperbolic tangent, see equation 3.32, can be used as well. Next, the output vector **h** of the hidden layer is passed through[5] the softmax function that produces the output vector of probabilities $\hat{\mathbf{y}}$. The sequence of these operations is called *forward propagation*.

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{3.32}$$

$$ReLU(x) = \max(0, x) \tag{3.33}$$

Since the MLP can generally contain more than one layer, we can rewrite the expression 3.31 representing the network in Figure 3.4 using the common notation [Jurafsky and Martin, 2009]:

$$\begin{aligned} \mathbf{z}^{(1)} &= \mathbf{W}^{(1)}\mathbf{a}^{(0)} + \mathbf{b}^{(1)} \\ \mathbf{a}^{(1)} &= g^{(1)}(\mathbf{z}^{(1)}) \\ \mathbf{z}^{(2)} &= \mathbf{W}^{(2)}\mathbf{a}^{(1)} + \mathbf{b}^{(2)} \\ \mathbf{a}^{(2)} &= g^{(2)}(\mathbf{z}^{(2)}) \\ \hat{\mathbf{y}} &= \mathbf{a}^{(2)} \end{aligned} \tag{3.34}$$

where the number in superscript refers to the n-th hidden layer starting from 1 and specifically the layer 0 means input, so $\mathbf{a}^{(0)}$ is the input vector $\mathbf{x}$, $g^{(i)}(\cdot)$ is the activation function in the i-ith layer, $\mathbf{a}^{(i)}$ is an output of the i-ith layer, $\mathbf{z}^{(1)}$ is the weighted sum in the i-ith layer and $\hat{\mathbf{y}}$ is the predicted output probability distribution. The activation function $g^{(2)}$ in the last (second) layer represents the softmax function in equation 3.31.

[5] In case of more than one layer, the output is passed into next hidden layer instead.

3.4.3 Learning of Neural Networks

As in the case of logistic regression, the goal is to optimize the parameters $\mathbf{W}^{(i)}$ and $\mathbf{b}^{(i)}$ in a way that the outputs $\hat{\mathbf{y}}$ for training data produced by the model are similar as much as possible to the true labels $\mathbf{y}$. The learning is done by the same *gradient descent* algorithm that was described in Section 3.3.4 and by the *back-propagation* algorithm [Rumelhart et al., 1986]. The same cross-entropy loss function (as for logistic regression, see equation 3.30), can be rewritten more comprehensively as follows:

$$\mathcal{L}_{CRE}(\hat{\mathbf{y}}, \mathbf{y}) = -\sum_{i=1}^{l} y_i \log \hat{y}_i \tag{3.35}$$

where l is number of classes, y_i is the gold label for the class i and $\hat{y}_i$ is the prediction of the model for the class i.

With the growing number of layers and parameters, the computation of the gradient becomes a complex and non-trivial task. The *back-propagation* algorithm [Rumelhart et al., 1986] allows computing the gradient. The back-propagation relies on the *chain rule*, given a composite function $f(x) = g(u(w(x)))$ the derivative $\frac{df}{dx}$ of the function $f(x)$ with respect to x is computed in a following way:

$$\frac{df}{dx} = \frac{dg}{du} \cdot \frac{du}{dw} \cdot \frac{dw}{dx} \tag{3.36}$$

Using the chain rule, the back-propagation algorithm computes the gradient composed of the partial derivative of the cost function with respect to each parameter of the model. The computed gradient is used by the gradient descent to update the parameters of the model as described in 3.3.4.

3.5 Recurrent Neural Network

The Recurrent Neural Network (RNN) [Elman, 1990] is intended for processing of sequential data. Text is sequential in nature. RNN allows processing sequences of different lengths, unlike the feed-forward neural network, where the input is always fixed-size. RNN also allows *"remember"* or *"persist"* information from the previous steps of the processed sequence because RNN takes as input not only the current input but also a hidden state of the network from the previous step, as shown in Figure 3.5.

More formally, RNN processes the input sequence $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2 \dots \mathbf{x}_T]$ and for each element $\mathbf{x}_t$ at time step t it computes new hidden state $\mathbf{h}_t$ from the input $\mathbf{x}_t$ and the previous hidden state $\mathbf{h}_{t-1}$. The new hidden state $\mathbf{h}_t$ is computed by hidden layer function $\mathcal{H}$:

$$\mathbf{h}_t = \mathcal{H}(\mathbf{x}_t, \mathbf{h}_{t-1}) \tag{3.37}$$

In the simplest case, the hidden layer function $\mathcal{H}$ is defined as:

$$\mathbf{h}_t = \sigma\left(\mathbf{W}_{xh}\mathbf{x}_t + \mathbf{W}_{hh}\mathbf{h}_{t-1} + \mathbf{b}_h\right) \tag{3.38}$$

where the $\mathbf{W}$ terms correspond to weight matrices (e.g., $\mathbf{W}_{xh}$ is the input-hidden weight matrix) and $\mathbf{b}_h$ term is hidden bias vector. The concrete implementation of the $\mathcal{H}$ function depends on the type of the used RNN unit [Graves et al., 2013], for example, Long Short-Term Memory (LSTM) unit [Hochreiter and Schmidhuber, 1997] or Gated Recurrent Unit (GRU) [Cho et al., 2014c]. Each RNN unit shares the parameters (weights) across all time steps. To propagate gradient in the gradient descent algorithm, the RNNs use the *back-propagation through time* [Werbos, 1988].

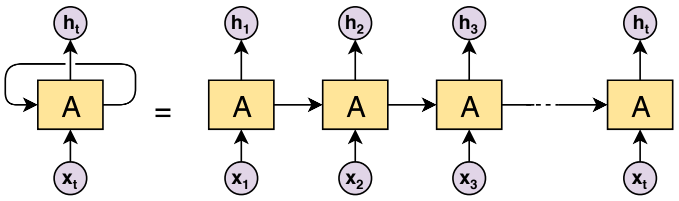


Figure 3.5: Basic RNN architecture[6].

The common practice is also to use *bidirectional* RNN (BiRNN) [Schuster and Paliwal, 1997]. BiRNN processes the sequence in both directions, which has shown to be beneficial because the output at time t can depend on the previous and also on future elements of the sequence. It is usually implemented with two RNN units, where one processes the sequence in the original order and the second RNN processes the sequence in reverse order. The outputs $\overrightarrow{\mathbf{h}_t}$ (for the original sequence direction, i.e., left to right) and $\overleftarrow{\mathbf{h}_t}$ (for the reversed direction) of these two RNN units are usually concatenated and producing one output $\mathbf{h}_t$ as follows:

$$\mathbf{h}_t = [\overrightarrow{\mathbf{h}_t}; \overleftarrow{\mathbf{h}_t}] \tag{3.39}$$

The disadvantage of BiRNN is that the entire input sequence must present when it is being preprocessed and it can be problematic for some specific tasks. An example of BiRNN is a bidirectional LSTM (BiLSTM) [Graves and Schmidhuber, 2005].

There is one common issue with the simplest RNN implementation (described above) called *vanishing* or *exploding gradients*. When RNN processes longer sequences during the training, the weights inside the RNN are multiplied in each time step. In the back-propagation step, there is also a large amount of multiplication and thanks to that, the gradients either *"explodes"* (become very large) or *"vanishes"* (become very small) and thus the model is not able to learn, i.e., instead of converging it diverges.

[6]Image based is on:
`http://colah.github.io/posts/2015-08-Understanding-LSTMs`

3.5.1 Long Short-Term Memory

Long Short-Term Memory (LSTM) [Hochreiter and Schmidhuber, 1997] is an implementation of the RNN capable and intentionally designed to *"remember"* or *"store"* some long-term dependency information from the previous time steps. The architecture of LSTM is shown in Figure 3.6.

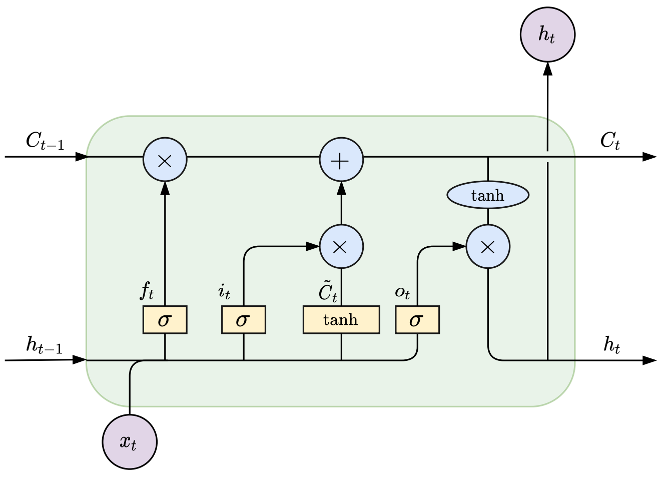


Figure 3.6: LSTM architecture visualisation[7].

The LSTM unit (or *cell*) is composed of structures called *gates* that can decide which information should be stored and which information should be deleted. The gates take the input $\mathbf{x}_t$ and previous hidden state $\mathbf{h}_{t-1}$ and produce output that is a part of the hidden state. In addition, the LSTM also takes the cell state $\mathbf{C}_{t-1}$ from the previous time step. More concretely, each LSTM unit contains *input*, *forget* and *output* gates. Using the gates, the input $\mathbf{x}_t$, previous hidden state $\mathbf{h}_{t-1}$ and previous cell state $\mathbf{C}_{t-1}$, the LSTM produces new hidden state $\mathbf{h}_t$ and new cell state $\mathbf{C}_t$. The entire model of LSTM can be written as follows:

$$
\begin{aligned}
\mathbf{f}_t &= \sigma\left(\mathbf{W}_{xf}\mathbf{x}_t + \mathbf{W}_{hf}\mathbf{h}_{t-1} + \mathbf{b}_f\right)\\
\mathbf{i}_t &= \sigma\left(\mathbf{W}_{xi}\mathbf{x}_t + \mathbf{W}_{hi}\mathbf{h}_{t-1} + \mathbf{b}_i\right)\\
\mathbf{o}_t &= \sigma\left(\mathbf{W}_{xo}\mathbf{x}_t + \mathbf{W}_{ho}\mathbf{h}_{t-1} + \mathbf{b}_o\right)\\
\tilde{\mathbf{C}}_t &= \tanh\left(\mathbf{W}_{xc}\mathbf{x}_t + \mathbf{W}_{hc}\mathbf{h}_{t-1} + \mathbf{b}_c\right)\\
\mathbf{C}_t &= \mathbf{f}_t \odot \mathbf{C}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{C}}_t\\
\mathbf{h}_t &= \mathbf{o}_t \odot \tanh\left(\mathbf{C}_t\right)
\end{aligned}
\tag{3.40}
$$

where $\odot$ is element-wise multiplication, $\mathbf{x}_t$ is the current input vector, $\mathbf{W}$ terms correspond to weight matrices and $\mathbf{b}$ terms are bias vectors, $\mathbf{i}_t$, $\mathbf{f}_t$, $\mathbf{o}_t$ are the outputs of the input, forget and output gates, respectively, $\tilde{\mathbf{C}}_t$ is

[7]Image based is on:
https://colah.github.io/posts/2015-08-Understanding-LSTMs.

the vector of new candidate information that can be added and σ is sigmoid function.

The forget gate $\mathbf{f}_t$ decides what information will be removed from the cell state. It produces values between 0 and 1 that are multiplied with the values from $\mathbf{C}_{t-1}$. The amount of information that will be deleted is controlled by the output where 0 means forgot (drop, delete) the entire information from the previous cell state $\mathbf{C}_{t-1}$ and 1 do not delete anything.

Next, the input gate $\mathbf{i}_t$ decides which values will be updated and $\tilde{\mathbf{C}}_t$ computes the new candidates' values (information) that could be potentially added. These two vectors are then point-wise multiplied together and the result is summed with $\mathbf{f}_t \odot \mathbf{C}_{t-1}$ that gives the new cell state $\mathbf{C}_t$. Finally, the new hidden state $\mathbf{h}_t$ is computed from the output gate $\mathbf{o}_t$ and the new cell state $\mathbf{C}_t$.

Another slightly simplified variant of LSTM is called Gated Recurrent Unit [Cho et al., 2014c].

3.6 Sequence to Sequence

Specific neural network architectures can also be used to model and solve *sequence* problems [Sutskever et al., 2014]. Sequence problems can be divided into four categories: *One-to-One*, *One-to-Many*, *Many-to-One* and *Many-to-Many*, as shown in Figure 3.7. The *many-to-many* sequence problem is also called *sequence to sequence* or *seq2seq*.

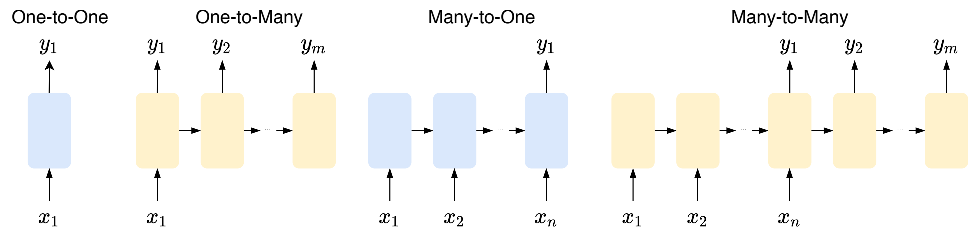


Figure 3.7: Sequence problems visualization[8].

We summarize the four categories as follows:

1. **One-to-One:** The *one-to-one* problem can be seen as a special case of sequence problem, where there is only one input and one output[9] with a fixed size. The example is image classification, where the input is always an image with a fixed size and the output is one category. Since text is in nature sequential, there are not many typical examples of one-to-one problems in NLP. A common text classification problem

[8]Image is based on `http://karpathy.github.io/2015/05/21/rnn-effectiveness`.
[9]By one input we mean one vector of features.

could be potentially considered as one-to-one because the problem used to be treated as one-to-one problem. The reason is that the input document (sequence) was transformed into one feature vector with fixed size regardless of the length of the document and then the document was classified with one category.

2. **One-to-Many:** There is only one input and the output is a sequence. For example, the input can be one word representing some topic and the output a sequence of words (sentences) about the topic. Another example is generating a description (sequence of words) for an image.

3. **Many-to-One:** In many-to-one problems, there is only a single output for the input sequence. Nowadays, a typical example is text classification, for example, sentiment polarity detection, where the input is a sequence of words and the output is a single polarity label.

4. **Many-to-Many:** Many-to-many or sequence to sequence problems consist of sequence input and sequence output, the lengths of the input and output sequences may differ. The most typical example is machine translation, where the input can be a sentence in Czech and the output is its translation in English. Another example is question answering.

3.6.1 Encoder-Decoder

The *many-to-one* and *many-to-many* problems are the most common in NLP. The common practice for modeling the *many-to-many* sequence problems in NLP is to use the *encoder-decoder* architecture [Cho et al., 2014c, Sutskever et al., 2014], see Figure 3.8 for basic visualization and Figure 3.9 for machine translation example implemented by RNN.

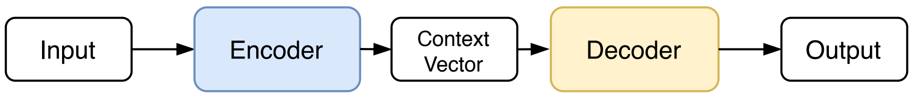


Figure 3.8: Basic visualisation of the encoder-decoder architecture.

The *encoder* encodes the variable-length input sequence into the fixed-length vector representation **C** (the inner state representing the input, also called context vector) and the *decoder* decodes the fixed-length vector representation **C** and generates the output, see [Cho et al., 2014c] for more detailed mathematical description. The outputs of the encoder are discarded.

The encoder-decoder architecture is usually implemented by RNN (see Section 3.5) or by the transformer model (see Section 3.7). Typically, the encoder and decoder are implemented by the same type of neural network. Both the encoder and decoder can be composed of multiple stacked layers of neural networks.

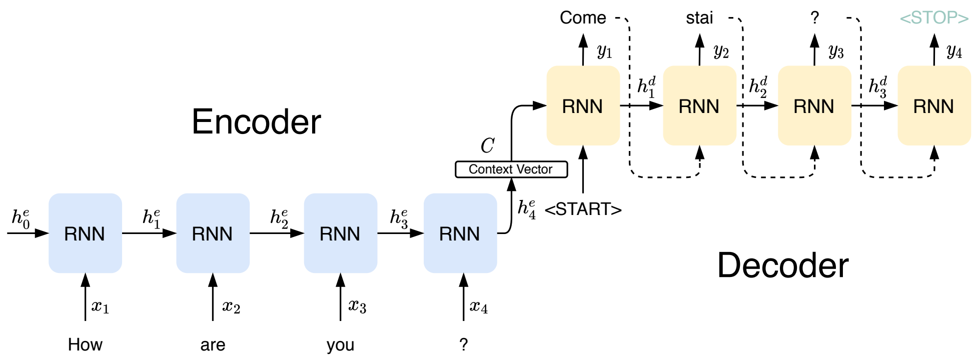


Figure 3.9: The example of encoder-decoder architecture for machine translation. The input sentence is encoded into context vector **C**. The output sentence is generated until the end of sentence tag <STOP> is generated.

The simplest solution for obtaining the context vector **C** is to use the last hidden state $\mathbf{h}_n^e$ of the encoder and the context vector **C** is then used as the initial hidden state $\mathbf{h}_0^d$ of the decoder [Jurafsky and Martin, 2009], that can be written as follows:

$$\begin{aligned} \mathbf{C} &= \mathbf{h}_n^e \\ \mathbf{h}_0^d &= \mathbf{C} \end{aligned} \tag{3.41}$$

During the generation of the output sequence, the hidden state $\mathbf{h}_t^d$ and the output probability distribution $\mathbf{y}_t$ at time step t is given by:

$$\begin{aligned} \mathbf{h}_t^d &= \mathcal{H}(\mathbf{y}_{t-1}, \mathbf{h}_{t-1}^d) \\ \mathbf{y}_t &= \text{softmax}(\mathbf{h}_t^d) \end{aligned} \tag{3.42}$$

where the function $\mathcal{H}$ represents the RNN cell. Eventually, the generation at each time step t can be conditioned by the context vector **C** and the previous output $\mathbf{y}_{t-1}$ as follows:

$$\begin{aligned} \mathbf{h}_t^d &= \mathcal{H}(\mathbf{y}_{t-1}, \mathbf{h}_{t-1}^d, \mathbf{C}) \\ \mathbf{y}_t &= \text{softmax}(\mathbf{h}_t^d, \mathbf{y}_{t-1}, \mathbf{C}) \end{aligned} \tag{3.43}$$

3.6.2 Attention Mechanism

The vanilla approach of the encoder-decoder architecture uses the hidden state of the last unit (the last RNN cell in encoder in Figure 3.9) as the context vector **C**. This solution is not optimal since the context vector is the last hidden state $\mathbf{h}_n$ of the encoder and the decoder is forced to produce the output using only the last state. Thus, the last state (a static and fixed length vector) must contain all necessary information about the entire source sequence. It is problematic in case of long dependencies where the information from the beginning of the sequence can fade away but can be important to produce the output at the end of the sequence.

The *attention mechanism* [Bahdanau et al., 2015] allows to model the long dependencies without regard to their distance in the input or output sequences [Vaswani et al., 2017]. The attention mechanism allows to the decoder use all hidden states of the encoder and also learn their importance (i.e., *"pay attention"*) in order to produce the output at the current time step. More formally, let us define the hidden state $\mathbf{h}_t^d$ of decoder at time step t as follows:

$$\mathbf{h}_t^d = \mathcal{H}(\mathbf{y}_{t-1}, \mathbf{h}_{t-1}^d, \mathbf{C}_t) \tag{3.44}$$

The only difference is that we replaced the static context vector $\mathbf{C}$ with a new vector $\mathbf{C}_t$ that is dynamically generated at each time step t. The architecture of encoder-decoder architecture with the attention mechanism is shown in Figure 3.10.

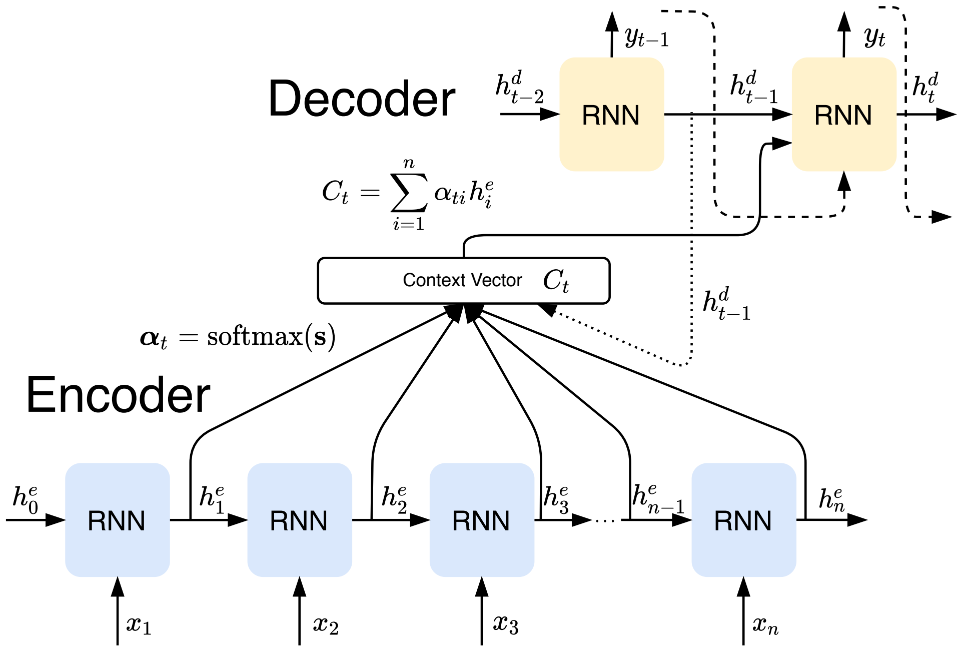


Figure 3.10: The example of encoder-decoder architecture with the attention mechanism.

The context vector $\mathbf{C}_t$ is computed from the previous hidden state $\mathbf{h}_{t-1}^d$ of the decoder and from an encoder hidden state $\mathbf{h}_i^e$ for each i, ($\forall i \in e$, e is a set of hidden states of the encoder). Firstly, set of scalars $score(\mathbf{h}_{t-1}^d, \mathbf{h}_i^e)$ is computed for the previous hidden state and all states $\mathbf{h}_i^e$ of the encoder using the following expression:

$$score(\mathbf{h}_{t-1}^d, \mathbf{h}_i^e) = \mathbf{h}_{t-1}^d \mathbf{W}_s \mathbf{h}_i^e \tag{3.45}$$

where $\mathbf{W}_s$ is a set of weights that is learned along with other parameters of the entire model. The resulting number (score) for each pair denotes the similarity between the two vectors. All scores are put into a vector $\mathbf{s}$ that is normalized with a softmax function and produces vector $\boldsymbol{\alpha}_t$ containing weights α_{ti}:

$$\boldsymbol{\alpha}_t = \text{softmax}(\mathbf{s}) \tag{3.46}$$

Each element α_{ti} of the vector gives us the relative importance of each encoder hidden state $\mathbf{h}_i^e$ to the current decoder state $\mathbf{h}_t^d$ [Jurafsky and Martin, 2009].

The context vector $\mathbf{C}_t$ for the decoder hidden state at time step t is given by weighting each encoder hidden state $\mathbf{h}_i^e$ by its corresponding weight α_{ti} as follows:

$$\mathbf{C}_t = \sum_{i=1}^{|e|} \alpha_{ti} \mathbf{h}_i^e \tag{3.47}$$

3.7 Transformer

Another very recent type of neural network architecture is called *transformer*, originally introduced in [Vaswani et al., 2017]. Most of the recent *generalized language models* like BERT or GPT 2 (see Chapter 4) use transformer as a basic building block. These models can achieve state-of-the-art results in a variety of NLP tasks proving transformer architecture to be very useful and promising. The transformer architecture is shown in Figure 3.11. Transformer is able to handle long dependencies in sequences using attention mechanism without any RNN. The advantage of transformer architecture is that unlike LSTM, the transformer can be easily parallelized.

3.7.1 Transformer Model

The transformer follows the encoder-decoder principle. For a given input sequence $(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n)$, the encoder produces a continuous representation $\mathbf{Z} = (\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_n)$. Then, the decoder generates an output sequence $(\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_m)$.

The encoder (left part of Figure 3.11) of the transformer consists of N identical layers[10] that are stacked on each other (N can be seen as a hyperparameter of the model). Each layer is composed of another two sub-layers. The first sub-layer is a multi-head self-attention mechanism and the second sub-layer is a fully connected feed-forward neural network. The output of each sub-layer is added to a residual connection [He et al., 2016] vector and a layer normalization [Ba et al., 2016] is performed. All sub-layers in the model and also the input embeddings layers produce outputs with a dimension $d_{\text{model}} = 512$.

[10] In the original transformer paper [Vaswani et al., 2017] the authors used $N = 6$.

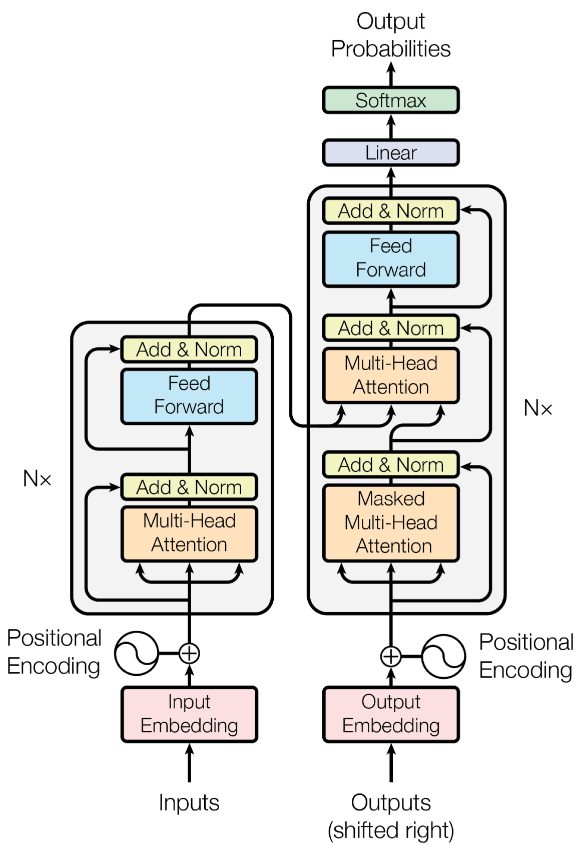


Figure 3.11: The architecture of the transformer model. The left part represents the *encoder* and the right part represents the *decoder*. Image is taken from [Vaswani et al., 2017].

The decoder (right part of Figure 3.11) of the transformer also contains N stacked layers identical to the encoder part besides the two following modification. (1) One extra sub-layer is added and it performs multi-head attention over the output of the encoder. (2) The first sub-layer (masked multi-head attention) is modified self-attention mechanism to ensure that the model predictions for position i depend only on the previous known outputs.

In each transformer layer block, the fully connected feed-forward network is applied to each sequence position separately. In each layer, the feed-forward neural network has its own parameters. It consists of two linear transformations with ReLU activation function between them. The feed-forward layer $\mathrm{FFN}(x)$ is given by:

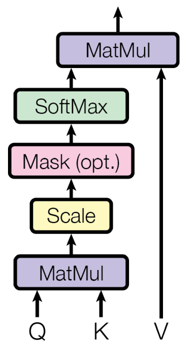
$$\mathrm{FFN}(\mathbf{x}) = \max(0, \mathbf{x}\mathbf{W}_1 + \mathbf{b}_1)\mathbf{W}_2 + \mathbf{b}_2 \tag{3.48}$$

The input sequence of the entire model is firstly used as an input for the two embeddings layers that produce vector representations for the tokens, the two embeddings layers share parameters. There is no information about the token position in the produced vector representation, thus the position is encoded and added to this vector. Each position is encoded with sine and cosine function, the produced positional encoding has the same dimension as the d_{model}.

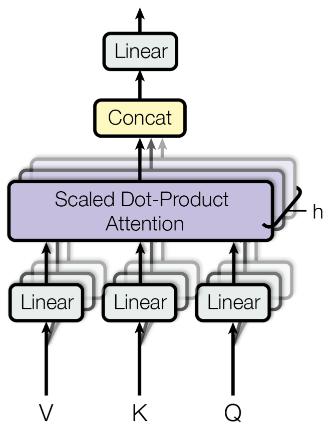
The output of the final decoder layer is passed to the learned linear transformation layer and then softmax function is used in order to produce the output tokens probabilities.

3.7.2 Self-Attention

Self-attention or *intra-attention* is an attention mechanism relating different positions of a single sequence to compute a representation of the same sequence [Vaswani et al., 2017]. It allows the model to learn dependencies and correlations between the current output at time step t and the previous parts of the sequence.



(a) Scaled Dot-Product Attention visualisation.



(b) Multi-head attention visualisation that consists of several attention layers running in parallel.

Figure 3.12: Attention visualisation. Images are taken from [Vaswani et al., 2017].

The attention mechanism used in the transformer is called *Scaled Dot-Product Attention*, see Figure 3.12a. This attention takes as an input *query* and *key* vectors of dimension d_k a *values* vector of dimension d_v. Firstly, the dot product of the query with all keys (i.e., size of the input sequence) is computed (the *MatMul* part in Figure 3.12a) and each of the computed dot products is divided (scaled) by $\sqrt{d_k}$. Next, the softmax function is applied[11]. The result is then multiplied with the *values* vector using the dot product.

In reality, the queries are stacked into a matrix $\mathbf{Q}$, the keys and values are also stacked into matrix $\mathbf{K}$ and $\mathbf{V}$, respectively. The computation is then

[11]In case of decoder the masking is also applied, see [Vaswani et al., 2017] for details.

done by matrix multiplication and the matrix of outputs can be written as follows:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^{\mathsf{T}}}{\sqrt{d_k}}\right)\mathbf{V} \tag{3.49}$$

3.7.3 Multi-Head Attention

Instead of computing the attention only once, multiple parallel attentions can be used. The attention is computed h times with separate parameters for each individual attention, where h denotes the number of heads (i.e., number of parallel attentions), see Figure 3.12b. The benefit of the multi-head attention is that it gives an opportunity to the model to learn a different type of information, for example, one attention could potentially learn to pay attention to syntax and the second could learn to pay attention to semantic information[12]. The outputs of the individual attentions are concatenated, linearly transformed and the result is passed to higher layers. The multi-head attention mechanism can be written as follows:

$$\begin{aligned} \text{MultiHead}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) &= \text{Concat}(\text{head}_1, \text{head}_2, \ldots, \text{head}_h)\mathbf{W}^O \\ &= [\text{head}_1, \text{head}_2, \ldots, \text{head}_h]\mathbf{W}^O \end{aligned} \tag{3.50}$$

where $\mathbf{W}^O \in \mathbb{R}^{hd_v \times d_{\text{model}}}$ is the matrix with parameters for the final linear transformation. The head head_i is given by:

$$\text{head}_i = \text{Attention}(\mathbf{Q}\mathbf{W}_i^Q, \mathbf{K}\mathbf{W}_i^K, \mathbf{V}\mathbf{W}_i^V) \tag{3.51}$$

where $\mathbf{W}_i^Q \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $\mathbf{W}_i^K \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $\mathbf{W}_i^V \in \mathbb{R}^{d_{\text{model}} \times d_v}$ are matrices with parameters corresponding to each head. The authors used $h = 8$ and $d_k = d_v = d_{\text{model}}/h = 64$.

3.8 Evaluation Metrics

The common evaluation metrics in SA are *accuracy*, F_1 *score* (or *F-measure*), *precision* and *recall*. Let us define some result cases that can occur during classification. The tested examples are classified into one of the possible classes[13] and based on the predicted class and the gold label (the actual true class of the example) they can be categorized into four types that: (1) *true positive* (*tp*) i.e., positive example was predicted as positive, (2) *false positive* (*fp*) i.e., negative example was predicted as positive, (3) *false negative* (*fn*) i.e., positive example was predicted as negative and (4) *true*

[12] This example is just to give you an idea, in reality, it does not have to be so clear.

[13] Here we consider binary classification, i.e., each example can be classified either as *positive* or *negative* but in general, any number of classes.

negative (tn) i.e., negative example was predicted as negative. Accuracy is a metric that summarizes the overall performance of the evaluated model and it is given by:

$$Accuracy = \frac{tp + tn}{tp + tn + fp + fn} \tag{3.52}$$

For a given class c, *precision* is the ratio of the number of correctly classified examples as class c to the total number of examples classified as class c. For a given class c, *recall* is the ratio of the number of correctly classified examples as class c to the total number of examples that are actually labeled with class c. Precision P and recall R for class c are computed as follows:

$$P^c = \frac{tp}{tp + fp} \tag{3.53}$$

$$R^c = \frac{tp}{tp + fn} \tag{3.54}$$

F_1 score for class c is computed with precision P^c and recall R^c as their harmonic mean, which is given by:

$$F_1^c = \frac{2 \times P^c \times R^c}{P^c + R^c} \tag{3.55}$$

The F-measure allows to specify parameter β that controls importance for precision and recall, then the F_β score for class c is computed as follows:

$$F_\beta^c = (1 + \beta^2) \times \frac{P^c \times R^c}{(\beta^2 \times P^c) + R^c} \tag{3.56}$$

In multiclass classification, the precision, recall and F-measure for each class can be *macro averaged*. The average metrics summarize the overall performance of the model. Macro recall R^M and macro precision P^M are computed as follows:

$$P^M = \frac{\sum_i^n P_i}{n} \tag{3.57}$$

$$R^M = \frac{\sum_i^n R_i}{n} \tag{3.58}$$

where n is a number of classes, P_i and R_i is precision and recall for individual classes. The macro F-measure is computed using P_i and R_i with formula 3.55. Usually, in classification, when the recall is improved, the precision drops and vice versa.

Chapter 4

Text Meaning Representation

Since machine learning algorithms require as an input vectors of numbers, the text must be converted and represented as vectors of numbers. Such a vector should be able to capture syntax and semantic information of the text.

Word embeddings (sometimes referred to as semantic space, word vectors or simply embeddings) have proven to be extremely useful in many NLP tasks and they became the essential part for most current NLP systems since they are able to capture meaning (semantic) of words. Probably the most famous methods for learning word embeddings are word2vec [Mikolov et al., 2013a], GloVe [Pennington et al., 2014] and fastText [Bojanowski et al., 2017]. They are based on the *Distributional Hypothesis* [Harris, 1954] that says that words that occur in the similar contexts tend to have similar meanings. It was popularized by [Firth, 1957] and his famous quote, "*a word is characterized by the company it keeps*".

4.1 Static Word Embeddings

Word embeddings (static word vectors) are typically pre-trained and represented by an n-dimensional vector space, also called semantic space. Each word w_i from vocabulary V is represented by a static vector $\mathbf{h}_i \in \mathbb{R}^n$ [Hewitt, 2019]. They are usually stored in one matrix which is used as a lookup table that maps words to vectors, it can be expressed as a mapping function:

$$f_{vocabulary} : w_i \rightarrow \mathbf{h}_i \tag{4.1}$$

We refer to them as *static word embeddings* (word2vec, GloVe, fastText[1]). The disadvantage of pre-trained static embeddings is that they cannot handle polysemy and do not consider the context. In other words, one word can have multiple meanings and the one concrete meaning depends on the

[1]Although fastText is able to compute vector for out-of-vocabulary words, we categorize it under static approaches.

context of the word. For example, consider these two sentences: "*I love Coca-Cola in the new can*" and "*I can buy Coca-Cola for you tonight*". In both sentences, the word *"can"* is used, but each time, depending on the context, it means something different and using the static word embeddings, it will be expressed with the same vector. The *dynamic word embeddings* are able to handle different contexts, see Section 4.2.

4.1.1 Similarity between Word Vectors

The similarity between words (word vectors) is usually measured with cosine similarity, which corresponds to the cosine of the angle α between the two vectors $\mathbf{x}$ and $\mathbf{y}$ with dimension d and it is computed as follows:

$$\cos(\mathbf{x},\mathbf{y}) = \frac{\mathbf{x}\cdot\mathbf{y}}{||\mathbf{x}||\ ||\mathbf{y}||} = \frac{\sum\limits_{i=1}^{d} x_i y_i}{\sqrt{\sum\limits_{i=1}^{d} x_i^2}\sqrt{\sum\limits_{i=1}^{d} y_i^2}} = \cos(\alpha) \tag{4.2}$$

4.1.2 Word2vec

The famous *word2vec* is a pair of two models for efficient learning of word embeddings, i.e., *Continuous bag-of-words* (CBOW) and *Skip-gram* proposed by Mikolov et al. [2013a]. Both word2vec models (CBOW and Skip-gram) are actually neural networks[2] with three layers: *input layer*, *projection layer* and *output layer*, see Figure 4.1. The models are inspired by the feed-forward neural network for language modeling (NNLM) proposed in [Bengio et al., 2003]. The proposed NNLM consists of input, projection, hidden and output layers, but the network is also computationally expensive, which is caused by the non-linear hidden layer. In word2vec, the hidden layer is removed and thus, the proposed models are computationally less expensive than the NNLM.

Skip-gram with Negative Sampling

Skip-gram model learns to predict the surrounding context words within a certain range C (context size) before and after the current word w_t as shown on the right side of Figure 4.1. The input of the model is only the current word w_t encoded with a one-hot vector and output is a probability distribution over a vocabulary V. The probability distribution denotes how likely the words will occur as context words around the word w_t.

The model is represented by an embedding matrix $\mathbf{X} \in \mathbb{R}^{|V|\times d}$ (between the *input* and *projection* layer) and a context embedding matrix $\hat{\mathbf{X}} \in \mathbb{R}^{|V|\times d}$

[2]Sometimes word2vec is not considered as a neural network because of the removed non-linearity that is so characteristic for neural networks.

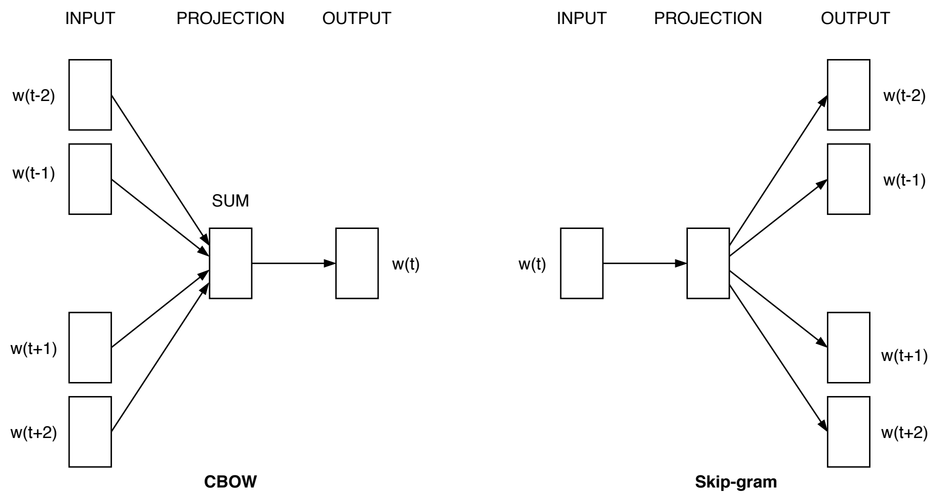


Figure 4.1: The CBOW architecture predicts the current word based on the context and the Skip-gram predicts surrounding words given the current word.

(between the *projection* and *output* layer), where d is specified dimension. These two matrices are parameters Θ (or weights) of the model. Given a sequence of training words $w_1, w_2, \ldots, w_T$, the model is optimized by minimizing the following objective function[3] $J_{SGNS}(\Theta)$:

$$J_{SGNS}(\Theta) = -\frac{1}{T}\sum_{t=1}^{T} \sum_{-C \leq j \leq C, j \neq 0} \log p(w_{t+j} \mid w_t) \tag{4.3}$$

and $p(w_{t+j} \mid w_t)$ is computed using the softmax function:

$$p(w_{t+j} \mid w_t) = \frac{\exp(\hat{\mathbf{x}}_{w_{t+j}}{}^{\mathsf{T}} \mathbf{x}_{w_t})}{\sum_{i=1}^{|V|} \exp(\hat{\mathbf{x}}_{w_i}{}^{\mathsf{T}} \mathbf{x}_{w_t})} \tag{4.4}$$

where $\hat{\mathbf{x}}_i$ is the context word vector from matrix $\hat{\mathbf{X}}$ for word w_i and $\mathbf{x}_i$ is the embedding word vector from matrix $\mathbf{X}$ for word w_i.

The original formulation of $\log p(w_{t+j} \mid w_t)$ is very expensive to compute (because of the denominator) and thus the *negative sampling* as an alternative solution for estimating the probability was proposed by Mikolov et al. [2013c]. The idea of the negative sampling is to help the model to distinguish the target word w_t from words (called negative samples) taken from a noise distribution $P_n(w)$. Words from the noise distribution $P_n(w)$ are unlikely to occur as the context words of the target word w_t. The noise distribution $P_n(w)$ was empirically estimated and set to the unigram distribution raised

[3]SGNS stands for *Skip-gram with negative sampling.*

to the 3/4th power. The usual number of negative samples is $5 - 20$. The negative sampling is given by:

$$p(w_{t+j} \mid w_t) = \log \sigma(\hat{\mathbf{x}}_{w_{t+j}}{}^{\intercal}\mathbf{x}_{w_t}) + \sum_{i=1}^{N} \mathbb{E}_{w_i \sim P_n(w)} \left[\log \sigma(-\hat{\mathbf{x}}_{w_i}{}^{\intercal}\mathbf{x}_{w_t})\right] \quad (4.5)$$

where N is a number of negative samples and σ is the sigmoid function. The definition of the sigmoid function is the same as in equation 3.14:

Continuous bag-of-words

The *continuous bag-of-words* architecture is similar to the Skip-gram architecture. The goal of the CBOW model is to predict the target center word w_i using all surrounding context words given by context size C. The CBOW objective function $J_{CBOW}(\Theta)$ to be minimized is defined as follows:

$$J_{CBOW}(\Theta) = -\frac{1}{T}\sum_{t=1}^{T} \log p(w_t \mid w_{t-C}, \ldots, w_{t-1}, w_{t+1}, \ldots w_{t+C}) \quad (4.6)$$

and $p(w_t \mid w_{t-C}, \ldots, w_{t-1}, w_{t+1}, \ldots w_{t+C})$ is defined as:

$$p(w_t \mid w_{t-C}, \ldots, w_{t-1}, w_{t+1}, \ldots w_{t+C}) = \frac{\exp(\hat{\mathbf{x}}_{w_t}{}^{\intercal}\overline{\mathbf{x}}_{w_t})}{\sum_{i=1}^{|V|} \exp(\hat{\mathbf{x}}_{w_i}{}^{\intercal}\overline{\mathbf{x}}_{w_t})} \quad (4.7)$$

where the vector $\overline{\mathbf{x}}_{w_t}$ is the sum of vectors of context words $w_{t-C}, \ldots, w_{t-1}, w_{t+1}, \ldots w_{t+C}$ defined as follows:

$$\overline{\mathbf{x}}_{w_t} = \sum_{-C \le i \le C, i \ne 0} \mathbf{x}_{w_{t+j}} \quad (4.8)$$

The negative sampling technique can be used for the CBOW as well. According to experiments presented in [Mikolov et al., 2013a], the CBOW model works slightly better than the Skip-gram model in capturing syntactic information, but the Skip-gram significantly outperforms the CBOW model in capturing semantic information.

4.1.3 Global Vectors

Global Vectors (GloVe) [Pennington et al., 2014] is a method for learning word vectors using co-occurrences of words in a training corpus. As first, the word co-occurrence matrix $\mathbf{C} \in \mathbb{R}^{|V| \times |V|}$ is constructed, where V is vocabulary and $|V|$ its size. Each entry $\mathbf{c}_{ij}$ of the matrix $\mathbf{C}$ denotes the number of times word w_j occurs in the context of word w_i. GloVe minimize the following objective function $J_{GloVe}(\Theta)$:

$$J_{GloVe}(\Theta) = \sum_{i,j=1}^{|V|} f(c_{ij})(\mathbf{x}_{w_i}{}^{\mathsf{T}}\hat{\mathbf{x}}_{w_j} + b_i + b_j - \log c_{ij})^2 \tag{4.9}$$

where $\mathbf{x}_{w_i} \in \mathbb{R}^d$ is a word vector of the word w_i, $\hat{\mathbf{x}}_{w_j} \in \mathbb{R}^d$ is a context vector of the word w_j, d is predefined dimensions of the vectors and f is a weighting function that should be relatively small for large values of c_{ij}, so that frequent co-occurrences are not overweighted and the function should be non-decreasing so that rare co-occurrences are not overweighted. The authors defined the function f as:

$$f(x) = \begin{cases} \left(\frac{x}{x_{max}}\right)^{\alpha} & \text{if } x < x_{max} \\ 1 & \text{otherwise} \end{cases} \tag{4.10}$$

where α and x_{max} are hyper-parameters, Pennington et al. [2014] used 3/4 for α and 100 for x_{max}. The function f is visualized in Figure 4.2.

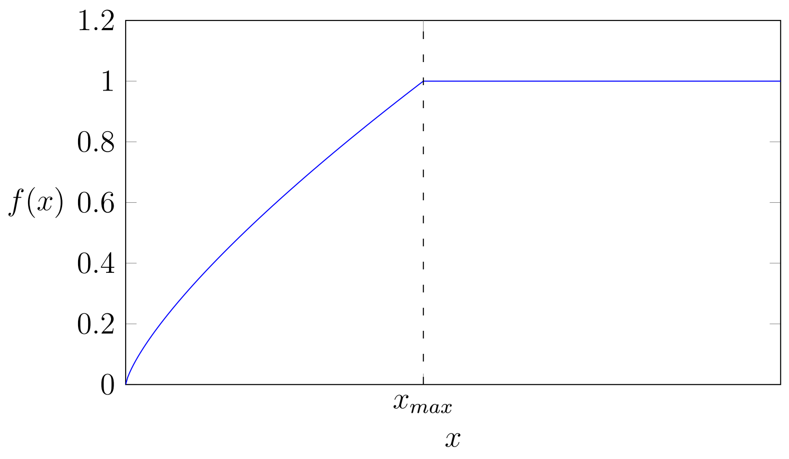


Figure 4.2: Weighting function f with $\alpha = 3/4$.

4.1.4 FastText

FastText [Bojanowski et al., 2017] is based on Skip-gram model with negative sampling and using sub-words (character n-grams). FastText uses character n-grams (hereinafter only n-grams) instead of entire words for training. Each n-gram has its own vector representation, the vector representation of words is computed as a sum of its character n-grams. Employing the sub-word information allows to improve the vector representation for morphologically rich languages. The advantage of this approach is that the model is able to obtain vector representation even for words that did not appear in the training corpus.

4.1.5 Sentiment Specific Word Embeddings

The static word embeddings pre-trained on general text (e.g., Wikipedia or news articles) will put words that occur in a similar context close to each other in the embeddings space, but some of them can have the opposite sentiment polarity. For example, the words *"good"* and *"bad"* often occur in a similar context and thus, in the semantic space, their vectors are similar as well. This feature for some specific sentiment words becomes the disadvantage in SA tasks because we would like to have these vectors not similar (as they have opposite sentiment orientation) and thus, the subsequent sentiment classifier will be able to use them for learning (distinguish between them). Such specialized word vectors are called *sentiment specific word embeddings* (SSWE).

The SSWE usually modifies the original semantic space to make the opposite sentiment words far from each other. Some deep learning models can do this modification implicitly without any explicit help. Such an example can be found in [Kim, 2014], where the word embeddings are used for initialization of input weights and the model then updates these weights during training and thus also modifying word embeddings.

The explicit methods for modifying word embeddings can be found in [Maas et al., 2011, Tang et al., 2014, Zhou et al., 2015, Barnes et al., 2018, Yu et al., 2017, Shi et al., 2018]. Maas et al. [2011] as first explored the idea of SSWE. They proposed a model for capturing sentiment and semantic using topic modeling approach similar to Latent Dirichlet Allocation [Blei et al., 2003]. Tang et al. [2014] used tweets labeled by distant-supervision. They extended the approach from [Collobert et al., 2011] using the hinge loss and introducing a second objective that is focused on a polarity of tweets. Zhou et al. [2015] proposed a method for learning bilingual sentiment word embeddings specifically for cross-lingual sentiment polarity classification. Barnes et al. [2018] also introduced bilingual sentiment word embeddings that jointly represent sentiment information in a source and target language. Another method was introduced in [Yu et al., 2017] that is based on refining already pre-trained word embeddings so that words can be closer to both semantically and sentimentally similar words and further away from sentimentally dissimilar words.

4.2 Dynamic Word Embeddings

In recent years, a new type of word embeddings (text representations) based on language models appeared. For example, *Embeddings from Language Models* (ELMo) [Peters et al., 2018], *Bidirectional Encoder Representations from Transformers* (BERT) [Devlin et al., 2019], *Universal Language Model Fine-tuning* (ULMFiT) [Howard and Ruder, 2018], *Generative Pre-Training* (GPT-2) [Radford et al., 2019] and *Generalized Autoregressive Pre-training*

(XLNet) [Yang et al., 2019], we call them *dynamic* or *contextualized word vectors* or *contextualized embeddings*[4]. The mentioned models are sometimes called *Generalized Language Models*.

The dynamic embeddings are functions of an entire sequence of text, unlike the static embeddings where the input is only one word. These models take into account not only the word itself but also the context of the word and thus, they eliminate the polysemy problem from static word embeddings. We can express such a model with the following function:

$$f_{context} =: (w_1, w_2, \dots, w_N) \to (\mathbf{h}_1, \mathbf{h}_2, \dots, \mathbf{h}_N) \tag{4.11}$$

where $(w_1, w_2, \dots, w_N)$ is a text sequence of words, $(\mathbf{h}_1, \mathbf{h}_2, \dots, \mathbf{h}_N)$ are vectors for the corresponding words and each word $w_i \in V$. In this case, the word embeddings are not pre-trained, but they are produced by a pre-trained model when they are needed.

In order to train them, usually, a large amount of unlabeled text and computational resources are needed. The individual models differ in architecture and in many other aspects, but they have in common two main properties. (1) They are intended to learn and build a strong contextual representation of language. (2) They are trained on objectives similar or closely related to language modeling [Hewitt, 2019]. The disadvantage compared to static word embeddings is that models for dynamic embeddings need much more computational resources to be trained.

These models are usually not explicitly trained for one particular NLP task (although there are exceptions, e.g., DocBERT [Adhikari et al., 2019] for document classification), but they are meant to capture the general representation of language. There are existing models for specific domains like biomedical texts *BioBERT* [Lee et al., 2019] or scientific publications *SciBERT* [Beltagy et al., 2019], but most of researchers use pre-trained and ready to use general models. The reason is that training of such a model is not trivial and it requires a large amount of computational resources and data. The pre-trained model is then fine-tuned on a specific downstream task.

Even though that the dynamic embeddings outperforms the static, the static embeddings are still used. The reason is that they can be easily trained for any language or domain, unlike the dynamic embeddings.

4.2.1 ELMo

Embeddings from Language Models (ELMo)[Peters et al., 2018] is a language model pre-trained with multiple layers of Bidirectional LSTM (BiLSTM). The language model is trained from text using both directions (left to

[4]Often they can represent the entire text sequence (instead of one word), e.g., sentence. Even though they can represent text sequence, we will refer to them as contextualized word vectors or word embeddings.

right and right to left) and it is called the *bidirectional language model.* The language representation is then stored in the internal states of the trained model. The word vectors can then be obtained by combining and weighting the weights or just by using any of the layers depending on the task that is being solved. The reason for weighting the layers or selecting one specific layer is that empirically was shown that the lower layer of the model is better at capturing syntax information and the higher layer is better at capturing context and semantic information.

More formally, given a sequence of n tokens $(x_1, x_2, \ldots, x_n)$ the classical (forward) language model learns to estimate the probability of the sequence by computing the probability of next token x_k given the history $(x_1, \ldots, x_{k-1})$. The probability of the sequence is then expressed as:

$$p(x_1, x_2, \ldots, x_n) = \prod_{k=1}^{n} p(x_k \mid x_1, x_2, \ldots, x_{k-1}) \tag{4.12}$$

The bidirectional language model uses the forward pass over the text and also the backward pass, where history (or rather following tokens) is given by the tokens after the target token which can be expressed as:

$$p(x_1, x_2, \ldots, x_n) = \prod_{k=1}^{n} p(x_k \mid x_{k+1}, x_{k+2}, \ldots, x_n) \tag{4.13}$$

The ELMo model consists of one input character-based and context-independent layer followed by L stacked BiLSTM layers[5], see Figure 4.3.

For each token x_k a set of $2L+1$ representations is computed. Each input token x_k is represented by one context-independent vector $\mathbf{x}_k^{LM}$ (which comes from the input layer) and then in each layer ℓ by two hidden states of the LSTMs $\overrightarrow{\mathbf{h}}_{k,\ell}^{LM}$, $\overleftarrow{\mathbf{h}}_{k,\ell}^{LM}$ for forward and backward direction, respectively. The overall representation R_k for the token x_k can be written as:

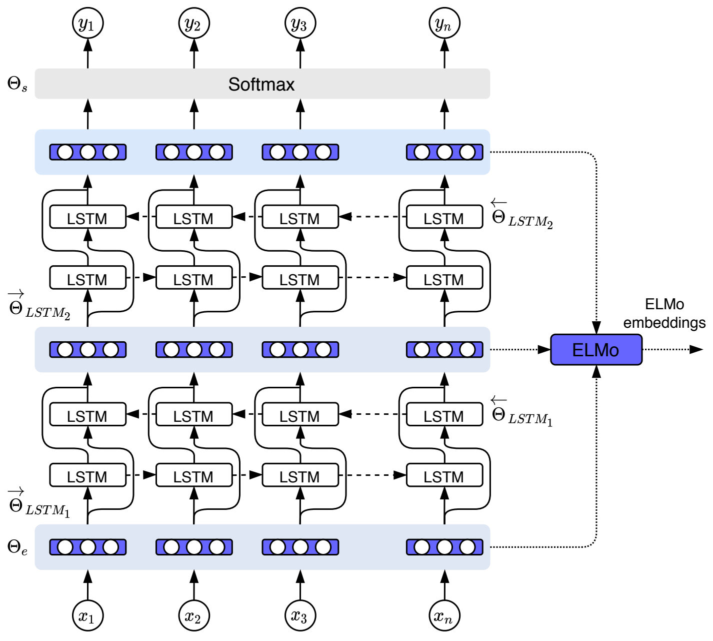
$$R_k = \{\mathbf{x}_k^{LM}, \overrightarrow{\mathbf{h}}_{k,\ell}^{LM}, \overleftarrow{\mathbf{h}}_{k,\ell}^{LM} \mid \ell = 1, \ldots, L\} = \{\mathbf{h}_{k,\ell}^{LM} \mid \ell = 0, \ldots, L\} \tag{4.14}$$

where $\mathbf{h}_{k,0}^{LM}$ is the representation from the first layer and $\mathbf{h}_{k,\ell}^{LM} = \left[\overrightarrow{\mathbf{h}}_{k,\ell}^{LM}, \overleftarrow{\mathbf{h}}_{k,\ell}^{LM}\right]$ is the representation of the higher layers. During the training the final layer representation $\mathbf{h}_{k,L}^{LM} = \left[\overrightarrow{\mathbf{h}}_{k,L}^{LM}, \overleftarrow{\mathbf{h}}_{k,L}^{LM}\right]$ is used as an input for a final softmax layer which outputs probabilities over tokens. The ELMo model is learned by minimizing the negative log-likelihood of the forward and backward directions:

$$\begin{aligned} J_{ELMo}(\Theta) = -\sum_{k=1}^{n} & (\log p(x_k \mid x_1, x_2, \ldots, x_{k-1}; \Theta_x, \overrightarrow{\Theta}_{LSTM}, \Theta_s) \\ & + (\log p(x_k \mid x_{k+1}, x_{k+2}, \ldots, x_n; \Theta_x, \overleftarrow{\Theta}_{LSTM}, \Theta_s)) \end{aligned} \tag{4.15}$$

[5]Any number of layers can be used. The authors used two layers.

where $\overrightarrow{\Theta}_{LSTM}$ and $\overleftarrow{\Theta}_{LSTM}$ are separate parameters for each direction, Θ_x and Θ_s are learnable parameters of the input layer and the softmax layer, respectively. Θ_x and Θ_s parameters are shared across the directions.



y_1
y_2
y_3
y_n
Θ_s
Softmax
LSTM
$\overleftarrow{\Theta}_{LSTM_2}$
$\overrightarrow{\Theta}_{LSTM_2}$
ELMo embeddings
ELMo
$\overleftarrow{\Theta}_{LSTM_1}$
$\overrightarrow{\Theta}_{LSTM_1}$
Θ_e
x_1
x_2
x_3
x_n

Figure 4.3: Architecture of the ELMo model.

4.2.2 Generative Pre-training Transformer

Generative Pre-training Transformer (GPT) [Radford et al., 2018] share the same idea with ELMo, i.e., train a language model on a huge amount of text data, which brings strong contextualized language representation that can be used in downstream tasks. GPT is a multi-layer transformer decoder, see Section 3.7 for transformer architecture. GPT differs from ELMo in two aspects. (1) ELMo uses independently trained forward and backward multi-layer LSTMs (GPT uses transformers). (2) The usage of both models in downstream tasks. ELMo produces word vectors that are used as additional features for a custom model designed for one specific downstream task, while GPT model itself is fine-tuned for the specific task, see Section 4.2.5.

The training procedure consists of two steps. In the first step, they train the model on a large corpus of unlabeled text with standard language modeling objective. The second step is to fine-tune the model for one specific downstream task. Another difference in comparison with ELMo is that GPT learns the language model only with a forward pass over the data, while ELMo uses both directions, i.e., forward and backward.

The GPT architecture is based on the original transformer [Vaswani et al., 2017], but it is modified and it uses only the decoder part of the transformer architecture, which is called *Transformer decoder* [Liu et al., 2018b]. The proposed model stacks 12 layers of transformers followed by the final softmax layer that produces a distribution over the target tokens, see Figure 4.4 for the architecture overview.

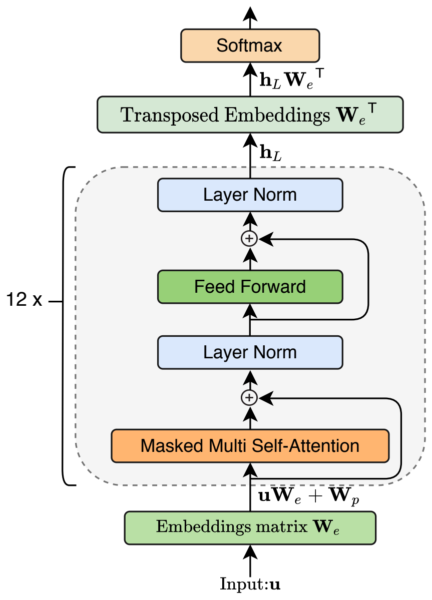

Softmax
$\mathbf{h}_L\mathbf{W}_e^\top$
Transposed Embeddings $\mathbf{W}_e^\top$
$\mathbf{h}_L$
Layer Norm
Feed Forward
Layer Norm
Masked Multi Self-Attention
12 x
$\mathbf{u}\mathbf{W}_e + \mathbf{W}_p$
Embeddings matrix $\mathbf{W}_e$
Input:$\mathbf{u}$

Figure 4.4: Architecture of the GPT model[6].

The objective function that is being minimized is similar to the one used in ELMo. For a given training sequence of tokens $\mathcal{X} = (x_1, x_2, \ldots, x_n)$ the model minimizes the following objective function $J_{GPT}(\mathcal{X}, \Theta)$:

$$J_{GPT}(\mathcal{X}, \Theta) = -\sum_{k=1}^{n} \log p(x_k \mid x_{k-j}, \ldots, x_{k-1}; \Theta) \tag{4.16}$$

where Θ are the optimized parameters of the model and j is the size of the context window. The model can be expressed as follows:

$$\begin{aligned} \mathbf{h}_0 &= \mathbf{u}\mathbf{W}_e + \mathbf{W}_p \\ \mathbf{h}_\ell &= \text{transformer\_block}(\mathbf{h}_{\ell-1}), \forall \ell \in [1, L] \\ p(V) &= softmax(\mathbf{h}_L \mathbf{W}_e{}^\top) \end{aligned} \tag{4.17}$$

where $\mathbf{h}_0$ is the hidden state (output) of the input layer, $\mathbf{u} = (u_{-j}, \ldots u_{-1})$ is a context vector of tokens, L is number of transformer layers[7], $\mathbf{h}_\ell$ are outputs of stacked transformer layers, $\mathbf{W}_e$ is the token embeddings matrix and $\mathbf{W}_p$ is

[6]Image is based on https://lilianweng.github.io/lil-log.
[7]The authors used 12 layers but any number could be used.

the position embeddings matrix and $p(V)$ is a probability distribution over the vocabulary V.

After the unsupervised training with the objective in Equation 4.16, the resulting pre-trained model can be fine-tuned for a certain task using supervision (i.e., task-specific labeled data). Given a training corpus $\mathcal{C}$, each training sample $c_i \in \mathcal{C}$ contains a sequence of tokens $(x_1, \ldots, x_m)$ along with a label y. The training sample is then passed through the pre-trained model and the output $\mathbf{h}_L$ of the last transformer layer is used as an input into a new softmax layer (with a matrix of parameters $\mathbf{W}_y$) that predicts the probability of label y:

$$p(y \mid x_1, \ldots, x_m) = softmax(\mathbf{h}_L \mathbf{W}_y) \tag{4.18}$$

The model is then trained (fine-tuned) by minimizing a new objective function $J_{GPT\_Supervised}(\mathcal{C}, \Theta)$ given by:

$$J_{GPT\_Supervised}(\mathcal{C}, \Theta) = -\sum_{k=1}^{n} \log p(y \mid x_1, \ldots, x_m; \Theta) \tag{4.19}$$

4.2.3 BERT

BERT stands for *Bidirectional Encoder Representations from Transformers* [Devlin et al., 2019]. It is a recent model for language representation based on transformer architecture.

Traditional methods for language modeling predict the next token in sequence using only the previous tokens or only tokens after the predicted token (they do not use both at the same time). Even some new methods, for example, GPT uses for learning only tokens before the target token. ELMo uses tokens before and after the target token, but it treats them independently. On the other hand, BERT is learning to predict the next token using the left context (previous tokens) and the right context (following tokens) jointly at once. This property is called *bidirectionality* (hence, the *bidirectional* word in BERT name), see Figure 4.5, and it leads to performance improvement in downstream tasks.

The authors of BERT proposed two models BERT$_{\text{BASE}}$ that has the same size as GPT (i.e., 12 stacked layers of transformer blocks, it contains 110 million parameters in total) in order to be comparable with GPT and BERT$_{\text{LARGE}}$ that consists of 24 stacked layers of transformer blocks with a total of 340 million parameters.

BERT training is similar to GPT, but it differs mainly due to the mentioned ability to learn jointly from both (backward and forward) directions when iterating over text. BERT is trained on a large unlabeled text corpus with two auxiliary tasks instead of the basic language task. (1) *Masked Language Modeling* (MLM), for a given input word sequence, a certain portion of words are replaced by a special symbol `[MASK]` and the goal of the task is

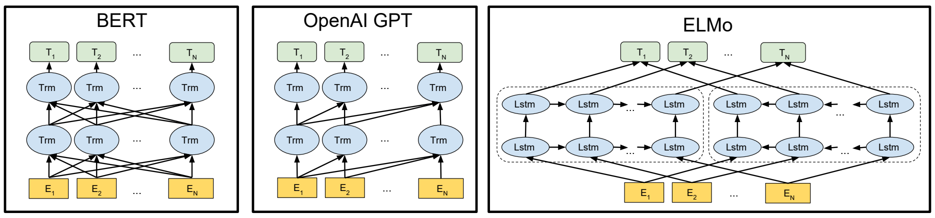


Figure 4.5: Comparison of pre-trained model architectures. BERT uses a bidirectional Transformer. GPT uses a left-to-right Transformer. ELMo uses the concatenation of independently trained left-to-right and right-to-left LSTMs to generate features for downstream tasks. Among the three, only BERT representations are jointly conditioned on both left and right contexts in all layers. Image was taken from [Devlin et al., 2019].

to recover the replaced original words without any information about them. (2) *Next Sentence Prediction* is a task where for a given sentence pairs A and B, the goal is to decide whether the B sentence follows the A sentence in a training corpus. The authors generated training corpus in that way that 50% of sentence pairs remained in the correct order and for 50% of sentence pairs, the B sentence was replaced by other random sentences from the corpus. Then the trained model is utilized for a specific downstream task in a similar way as GPT.

Since BERT model contains millions of parameters, the training is very computationally expensive. Thus, some modifications and variants of BERT were introduced to reduce the number of parameters and the training time, for example, ALBERT [Lan et al., 2020] or DistilBERT [Sanh et al., 2019].

4.2.4 XLNet

XLNet Yang et al. [2019] is the newest BERT-like pre-trained model for language representation. It outperforms BERT on 20 NLP tasks. XLNet is similar to BERT, it also learns bidirectionally, but in addition, it tries to overcome some limitations of BERT.

The main difference between BERT and XLNet is that XLNet is a generalized autoregressive (AR) method for language modeling. Conventional AR language models employ context to predict the next word in a sequence using either *forward* or *backward* context. It can not use forward and backward context at the same time, this is an obvious disadvantage because some downstream language understanding tasks often require context information from both directions.

On the other hand, BERT is referred to as *autoencoder* (AE) language model. The AE language models learn on predicting original words from a corrupted input text. The advantage of this approach is that it allows to learn jointly from both context directions (forward and backward). The disadvantage of BERT is its learning task called *Masked Language Modeling* (see the previous section). BERT uses the bidirectional context for a prediction of the original words and it leads to performance improvement. However, the `[MASK]` symbol is not present when the model is fine-tuned for a certain downstream task, resulting in a pretrain-finetune discrepancy. The second disadvantage caused by the masking is that if there is more than one masked word, then BERT assumes that the predicted words are independent, which is not true in many cases [Yang et al., 2019]. For example, the sentence "*Real Madrid is a Spanish football club*" can be masked as "`[MASK] [MASK]` *is a Spanish football club*". There is clear dependence between words *Real* and *Madrid*, but BERT ignores this dependency, unlike XLNet.

XLNet tackles this problem by introducing a new language modeling task, called *permutation language modeling*, where all tokens are predicted in a random order, see Yang et al. [2019] for a detailed description. This task used for learning, allowing the model to learn bidirectionally, unlike other traditional AR language models.

4.2.5 Transfer Learning and Fine-tuning

Transfer learning is a technique used in machine learning, especially in computer vision and NLP. The idea is to pre-train model on data (usually large unlabeled dataset) and then use and adapt this model to another specific task, i.e., transfer some knowledge. The detailed description and categorization of transfer learning techniques can be found in [Pan and Yang, 2010].

The motivation for this technique is that for most supervised NLP tasks, the labeled data are limited (insufficient amount of training data). Transfer learning allows us to use some general knowledge from the pre-trained model and thus improve performance on the task with limited data. The transferred general knowledge would not be possible to learn with the small amount of the available data for the specific task.

The mentioned models of dynamic embeddings (i.e., generalized language models like ELMo, BERT, GPT, XLNet etc.) are directly intended to be used as models for transfer learning. They can be categorized according to their usage in downstream tasks into two groups [Devlin et al., 2019]:

1. *Feature-based approach*: The pre-trained model produces vector representations of text and these representations are used as additional features for another custom model for a specific task. The ELMo architecture belongs under this category.

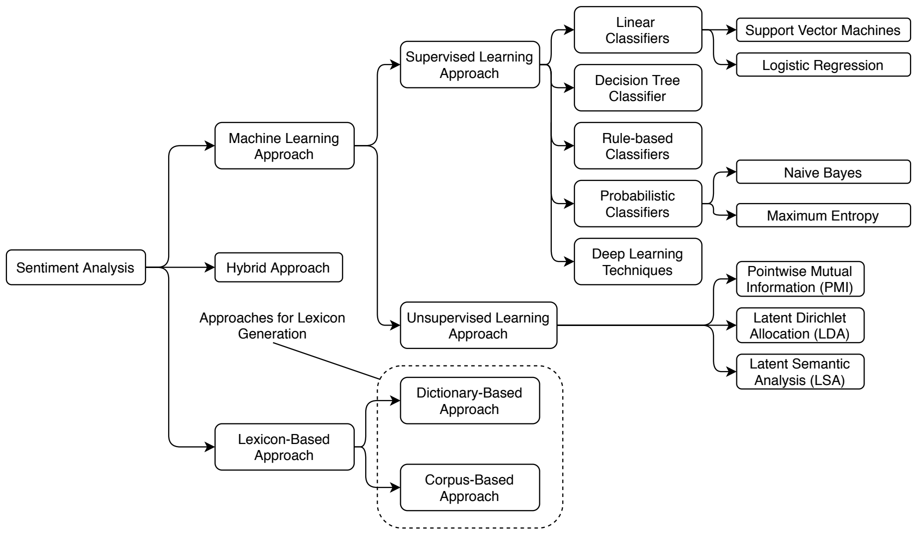
2. *Fine-tuning approach*: The pre-trained model (its parameters) is directly fine-tuned on the downstream task and no additional model is needed. Examples of this approach are BERT or GPT.

Chapter 5

Approaches for Sentiment Analysis

In the next sections, we describe common or basic approaches applicable for aspect sentiment classification, document-level and sentence-level polarity detection (also called sentiment analysis). At the end of this chapter we give an overview of state-of-the-art approaches.

In general, all approaches can be placed into three groups; *lexicon-based approach, machine learning approach* and *hybrid approach*. The machine learning approach can be further divided into *supervised learning approach* and *unsupervised learning approach* [Giachanou and Crestani, 2016, Liu, 2012, Medhat et al., 2014, Maynard and Funk, 2011] as is shown in Figure 5.1. Figure 5.1 shows the categorization of approaches that can be applied to SA and other related tasks.



Sentiment Analysis
Machine Learning Approach
Hybrid Approach
Lexicon-Based Approach
Supervised Learning Approach
Unsupervised Learning Approach
Approaches for Lexicon Generation
Dictionary-Based Approach
Corpus-Based Approach
Linear Classifiers
Decision Tree Classifier
Rule-based Classifiers
Probabilistic Classifiers
Deep Learning Techniques
Support Vector Machines
Logistic Regression
Naive Bayes
Maximum Entropy
Pointwise Mutual Information (PMI)
Latent Dirichlet Allocation (LDA)
Latent Semantic Analysis (LSA)

Figure 5.1: Sentiment analysis techniques overview.

5.1 Lexicon-based Approaches

The *lexicon-based* approaches use a list of predefined words or phrases, called *sentiment lexicons* or *opinion lexicons*. For each word or phrase, they contain sentiment orientation, which can be a real number denoting a sentiment score where positive value means positive sentiment and negative number means a negative sentiment [Liu, 2012, Singh et al., 2018, Giachanou and Crestani, 2016, Medhat et al., 2014, Bravo-Marquez et al., 2014]. These sentiment words values are then in each sentence, document or other pieces of text (based on their presence in the text) summed up and the resulting number decides the sentiment polarity of the document or sentence. The calculation of the final sentiment polarity can be done in many ways and does not necessarily have to be summation. The score is usually normalized, for example, by the length of the evaluated text.

The advantage of lexicon-based approaches is their easy implementation in case that we already have the sentiment lexicon. In current times, there is a large number of available lexicons for different languages and domains, see Section 2.7.1 for examples. A huge advantage of this approach is that there is no need for training data that can be extremely useful because, in practice, we may not have enough training data or even any data to train a supervised machine learning approach. The disadvantage is that some implementations of this approach can not handle negation (e.g., *not so good* or *not bad*) or contrary (e.g., *It is good, but I don't like it*) expressions which usually shift or change the sentiment orientation. Further text without sentiment words may also express sentiment. In such a case, the lexicon-based approach will fail.

The vast majority of lexicon-based approaches are currently obsolete and overcome by recent state-of-the-art methods based on supervised learning. However, in some cases in practice, there may not be any other applicable approach or it just makes sense to use the lexicon-based approach. For example, the *European Media Monitor* (EMM) system [Steinberger et al., 2017] from Joint Research Center[1] (JRC) uses a lexicon-based approach for SA in almost all European languages. Obtaining training data for so many languages to use the traditional supervised machine learning algorithms or even deep neural network would be enormously difficult and expensive.

5.1.1 Existing Lexicon-Based Methods

For the document-level SA, supervised machine learning techniques are usually used, but there are also works using lexicon-based methods. The Europe Media Monitor (EMM) system from Steinberger et al. [2017], Balahur et al. [2010], is also used to detect sentiment polarity (in the paper they call it

[1]Joint Research Center is a research center of European Commission

tonality) in news articles. The EMM system counts occurrences of language-specific sentiment terms from their language-specific dictionaries. Each sentiment term has a sentiment value assigned. The system sums up values for all words (which are present in the mentioned dictionary) in a given text. The resulting number is normalized and scaled to a range from -100 to 100 where the negative value indicates negative sentiment and the positive value indicates positive sentiment and the neutral sentiment is expressed with zero.

Balahur et al. [2009] detected sentiment quotations from newspaper articles. They used multiple sentiment lexicons to detect the sentiment. They mapped terms in the lexicons to a common score scale and they summed the scores of sentiment words that are present in the quotations. The polarity of each quotation was determined by the resulting summed value, i.e., a positive value means positive sentiment and a negative value means negative sentiment.

Cho et al. [2014a] classify product reviews. They tackle the problem of the lexicon domain dependency. They merge multiple sentiment dictionaries in order to expand the usable dictionary. Then, they modify the resulting lexicon to adapt it for a specific domain. They remove non-profitable words (words that do not improve the classification) from the dictionary or they change sentiment orientation of some words. The sentiment score of each review D_j is computed as follows:

$$score(D_j) = \frac{1}{n}\sum_{i=1}^{n} w_i \tag{5.1}$$

where w_i is a sentiment word present in the dictionary and in the review D_j and n is a number of matched dictionary words.

5.1.2 Lexicon Methods for Aspect-Based Sentiment

The early approach is used in [Hu and Liu, 2004], where the author uses WordNet [Miller et al., 1990] lexicon by counting positive and negative expressions.

A more advanced approach is described in [Ding et al., 2008] that takes into account context dependent sentiment words and negations. They focus on customer reviews of products and sentiment expressed towards their aspects, assuming that entities and aspects are known. Their main idea to determine the polarity towards a particular aspect is to use sentiment words around the product aspect. They compute a score for each aspect. For a given sentence s with a set of aspects A and list of all sentiment words V with their sentiment orientation O value, they compute the sentiment score for each aspect $a_j \in A$ using the following function:

$$score(a_j, s) = \sum_{w_i : w_i \in s \wedge w_i \in V} \frac{O_{w_i}}{dis(w_i, a_j)} \tag{5.2}$$

where w_i is the sentiment word, O_{w_i} is the sentiment orientation value of the sentiment word w_i and $dis(w_i, a_j)$ is the distance between aspect a_j and the sentiment word w_i. Thanks to this function, the sentiment words that are far away from the aspect a_j get lower weights. If the final score is negative, then the sentiment of the aspect a_i in sentence s is negative. If the final score is positive, then the sentiment of the aspect is positive. It is neutral otherwise. The sentiment orientation value of -1 is assigned to negative sentiment words and $+1$ to positive sentiment words. They handle negations and *but-clauses* separately. For the sentiment words that are close to negation words (e.g., *no, not, never, none, nobody etc*), they reverse the original sentiment orientation. The second case are the *but-clauses* which are words or phrases indicating *contrary* and they often change the sentiment orientation. The most common contrary word in English is the word "but" but there are also other phrases like "except that", "except for" or "with the exception of".

5.2 Supervised Learning Approaches

Supervised learning is the most common way to solve the SA task [Liu, 2012, Medhat et al., 2014], there is a huge amount of papers using supervised machine learning, for example, [Pang et al., 2002, Martineau and Finin, 2009, Go et al., 2009, Pak and Paroubek, 2010, Balahur and Turchi, 2012, Socher et al., 2013, Kiritchenko et al., 2014, Kim, 2014, Baziotis et al., 2017, Sun et al., 2019b] and many more. Usually, the task of SA is treated as a regular text classification problem, i.e., classify text into one of n predefined classes. But a regression task can be employed as well, in that case, the task is to predict continuous number (sentiment score) instead of one predefined class.

In supervised learning, there is a list of examples $X = \{x_1, x_2, \ldots, x_n\}$ for training (e.g., sentences, documents, tweets etc.). Each training record is labeled with one label, for example, *positive*, *negative* or *neutral*, but in general, any number of predefined classes can be used. The labels are given by a list $Y = \{y_1, y_2, \ldots, y_n\}$. The goal is to create a model using the training list X and a list of labels Y that predicts output y_i based on input x_i. The model represents function f, which map input X to output Y. It can be rewritten as mapping function f as follows:

$$f : X \to Y \tag{5.3}$$

During the training of the model, the goal is to find the best approximation of mapping function f.

5.2.1 Features for Supervised Learning in NLP

The input of supervised machine learning algorithms are vectors of numbers, but in NLP, there is usually only unstructured text. To be able to use text as the input for traditional machine learning methods, features from the text must be extracted and converted to numbers.

In the case of deep learning methods, the features are not selected and extracted manually, but the model itself will extract and learn which features are important and beneficial for the model from raw input.

Although the huge improvement in deep learning, traditional approaches are still used in practice. Next, we describe features for SA [Liu et al., 2010, Medhat et al., 2014, Giachanou and Crestani, 2016] used in the traditional supervised machine learning methods.

Terms Presence and Frequency

One of the most common and basic features is the presence of individual words (*unigrams*) or other *n-grams* with their frequency. The frequency value can be expressed in multiple ways, such as, binary (one if the word or n-gram is present, zero otherwise) or by a number of occurrences of the n-gram. Another option is to use a weighting method for the number of occurrences, for example, *tf-idf* weighting scheme [Manning et al., 2010] which is defined as follows:

$$\textit{tf-idf}_{t,d} = tf_{t,d} \times \textit{idf}_t \tag{5.4}$$

where $tf_{t,d}$ is a logarithmic term frequency and $\textit{idf}_t$ is an inverse document frequency. They are computed as follows:

$$tf_{t,d} = \begin{cases} 1 + \log cf_{t,d} & \text{for } cf_{t,d} > 0 \\ 0 & \text{otherwise} \end{cases} \tag{5.5}$$

$$\textit{idf}_t = \log \frac{N}{\textit{df}_t} \tag{5.6}$$

where $cf_{t,d}$ is a number of occurrences of term t in document d, N is the total number of documents in the training collection and $\textit{df}_t$ is a number of documents in the training collection in which the term t is present.

The idea of using any weighting method is that it takes into account the relative importance of n-grams in the training data. In general, any weighting scheme can be used. The common practice is to use n-grams that appeared in the training data at least n times. Apart from the word n-grams presence, character-based n-grams can be used as well.

Part of Speech

The part-of-speech (POS) tags of words can hold valuable information related to SA, such as adjectives. So the presence of certain POS tags, its combination or counts can be used as separate features. The n-gram representation of POS tags can also be used, the words are replaced by their corresponding POS tags and n-grams are created in the same way.

Sentiment Words and Phrases

Words, phrases and idioms from sentiment lexicons (see Section 2.7.1) can also be used as features. The presence of positive (*good, nice*) or negative (*bad, poor*) words can be a beneficial indicator of sentiment. Most sentiment words are adjective or adverbs, but verbs (e.g., *hate, like* or *love*) and nouns (e.g., *junk, rubbish*) hold some sentiment as well [Liu, 2012]. The feature can be constructed in many ways, for example, a binary presence of sentiment words from a certain lexicon, a ratio of positive/negative words or sum of sentiment scores of the sentiment words. These features are called *lexical features.*

Negations

Negations in text change the sentiment orientation. Thus their presence can be used as a feature.

Word Embeddings

Word in semantic space (also called word embeddings, See Chapter 4) is represented as a dense vector of real numbers. This vector represents the meaning (semantic) of the word. The traditional word embeddings are produced by word2vec [Mikolov et al., 2013a], GloVe [Pennington et al., 2014] and fastText [Bojanowski et al., 2017] algorithms. There are also embeddings called *sentiment specific word embeddings* (SSWE) which are created specially for SA tasks, for such example see [Tang et al., 2014].

In general, text (sentences, paragraphs or documents) vary in length and traditional supervised machine learning algorithms take as an input a fixed number of features (vector of fixed dimension) and they are not able to handle different feature input size (different number of input vectors in this case). The simplest method to obtain a single vector representation of any text is to use global pooling (e.g., average-pooling or max-pooling) over their word vectors. The resulting vector then has the same dimension and it is independent on the text length.

Syntactic Features

Apart from the POS tags, other syntactic features can be generated from parsing or dependency trees. These features capture word dependencies and structure of sentences.

Stylistic Features

Stylistic features [Giachanou and Crestani, 2016], are typically used for a text from social media websites (Facebook, Twitter). They capture some non-standard writing styles like emoticons, emojis, abbreviations, slang expressions or specific usage of punctuation.

5.2.2 Existing Supervised Methods

In this section, we present some important works for SA that use supervised machine learning. The recent state-of-the-arts methods are discussed in Section 5.5. Again, we do not distinguish between methods for the *document-level* and *sentence-level* polarity detection despite the fact that some of them were developed for a specific type of text and they may slightly vary when used on a different type of text.

Work presented in [Pang et al., 2002] is one of the earliest that uses supervised machine learning. They classified movie reviews as positive or negative with Naive Bayes classifier, SVM and Maximum Entropy classifier. They experimented with the following features: unigrams, bigrams, adjectives and POS tags.

Go et al. [2009] focused on the classification of Tweets with distant supervision using Naive Bayes classifier, Maximum Entropy classifier and SVM. The work was based on [Pang et al., 2002], they used similar features as well as similar machine learning algorithms. They created a training dataset of 1.6M tweets (50% negative and 50% positive) automatically.

Pak and Paroubek [2010] built a Naive Bayes classifier based on traditional n-gram and POS features that is able to classify Tweets as positive, negative or neutral.

Socher et al. [2013] introduced neural network architecture called *Recursive Neural Tensor Network* and nowadays very known *Stanford Sentiment Treebank* dataset. The model was able to capture accurately the effect of negation and its scope at various tree levels for both positive and negative phrases. They used a corpus of movie reviews from [Pang and Lee, 2005], they parsed the dataset's sentences into o parse trees, which were subsequently annotated by human judges. Using the novel Recursive Neural Tensor Network and the created dataset, they were able to push state-of-the-art result in sentence positive/negative classification from 80% of accuracy up to 85.4%.

Another breakthrough work is presented in [Kim, 2014]. He was first, who effectively used a *convolutional neural network* (CNN) and pre-trained word embeddings for polarity detection and other sentence-level classification tasks. The proposed model improved the state-of-the-art results on 4 out of 7 tasks, including SA.

Baziotis et al. [2017] won with their deep learning system the SemEval-

2017 Task 4 competition called *Sentiment Analysis in Twitter* [Rosenthal et al., 2017]. They employed Long Short-Term Memory (LSTM) network with attention mechanism, on top of pre-trained word embeddings (they classified Tweets as positive/negative/neutral).

The last cited papers [Socher et al., 2013, Kim, 2014, Baziotis et al., 2017] showed that incorporating the deep learning techniques and neural networks is beneficial, significantly outperforms the traditional supervised machine learning algorithms and push forward the state-of-the-art results in SA. We describe some of the very recent state-of-the art methods for SA in Section 5.5.

5.2.3 Supervised Aspect-Based Methods

As we mentioned in Section 2.3.3, the aspect-based SA is composed of several tasks. Here, we discuss some supervised methods for *Aspect sentiment classification* and *Aspect extraction*. To the *Aspect extraction* task, four main types of approaches[2] can be applied [Liu, 2012].

(1) Aspect extraction using *frequent nouns and noun phrases* relies on finding and counting of their occurrence frequencies. This method is applied to a large number of reviews in a specific domain. Hu and Liu [2004] use POS tags to identify nouns. The most frequent nouns and noun phrases are kept (considered as aspects) and the less frequent are removed. Improvement of this approach was proposed in [Popescu and Etzioni, 2005].

Another similar approaches can be found in [Blair-Goldensohn et al., 2008, Moghaddam and Ester, 2010, Scaffidi et al., 2007, Long et al., 2010, Zhu et al., 2009].

(2) Aspect extraction by *exploiting opinion and target relations*. Some referenced papers can be found in [Liu, 2012].

(3) *Supervised machine learning* can also be used to extract aspects. Since it is a supervised based method, it requires labeled data. The common methods are based on *sequential learning*, for example, *Hidden Markov Models* (HMM) [Rabiner, 1990] or *Conditional Random Fields* (CRF) [Lafferty et al., 2001]. The HMM approach was used in [Jin and Ho, 2009] and usage of CRF can be found for example, in [Jakob and Gurevych, 2010, Choi and Cardie, 2010, Hercig et al., 2016].

(4) *Topic models* are the last type. Topic modeling is an unsupervised method that models a probability distribution of topics in a document and a probability distribution of words for each topic. The output of topic modeling is a set of word clusters. The following works used topic models for aspect extraction: [Mei et al., 2007, Titov and McDonald, 2008, Brody and Elhadad, 2010, Li et al., 2010a, Zhao et al., 2010, Mukherjee and Liu, 2012,

[2]Only one approach is based on supervised learning, but we mention all four.

Xianghua et al., 2013]. Next, we mention more recent works focused on aspect extraction.

Poria et al. [2016] first used the deep learning approach to aspect extraction. They employed convolutional neural network in combination with linguistic patterns and word embeddings to tag each word in a sentence as either aspect or non-aspect word. [Shu et al., 2017] tries to tackle the problem of domain dependency. They proposed a method to use a pre-trained CRF model for aspect extraction on different domains to improve results on a new domain. In [Xu et al., 2018] is proposed novel convolutional neural network model with two types of pre-trained word embeddings, i.e., general-purpose embeddings and domain-specific embeddings. The usage of two types of embeddings brought performance improvement and the model outperformed the other state-of-the-art methods at that time.

In the case of the *aspect sentiment classification* task, supervised methods for sentence-level polarity detection can also be used. Recent models usually rely on neural networks. Khalil and El-Beltagy [2016] used CNN classifier with fine-tuned word embeddings for a specific domain to detect aspect sentiment polarity of laptops and restaurant reviews. Chen et al. [2017] proposed a novel model that adopts a multiple-attention mechanism to capture sentiment features separated by a long distance. They combined multiple attentions with a recurrent neural network, concretely Long Short-Term Memory and Gated Recurrent Unit. [Liu et al., 2018a] proposed a novel recurrent neural network architecture with external memory and with a delayed memory update mechanism to track entities specifically for the aspect-based SA task. This is one of the recent state-of-the-art approaches for this task. For a more detailed description of other tasks in aspect-based sentiment see [Liu, 2012, Do et al., 2019, Zhang et al., 2018].

5.3 Unsupervised Learning Approaches

Unlike the supervised learning, the *unsupervised machine learning* usually takes only input data $X = \{x_1, x_2, \ldots, x_n\}$ with no defined output (labels). The unsupervised algorithm then finds some patterns and structures in the input data with a minimal human supervision. The unsupervised techniques usually require much more data than the supervised techniques. There is also *semi-supervised learning*, which is between the unsupervised approaches and supervised approaches and usually, only a small part of the input data X is labeled and labels are missing for the rest of the input data.

The supervised approach is much more common for SA, although unsupervised methods are studied as well. [Turney, 2002] is one of the earliest works in SA and the unsupervised approach was there used to classify reviews as *recommended* (positive) or *not-recommended* (negative). In the first step, they use a POS tagger to identify phrases of adjectives or adverbs. In the second step, they estimate the sentiment orientation SO of the phrase composed of words w_1 and w_2 using *pointwise mutual information* (PMI) which is computed as follows:

$$PMI(w_1, w_2) = \log_2 \left(\frac{p(w_1 \wedge w_2)}{p(w_1) \times p(w_2)} \right) \tag{5.7}$$

where $p(w_1 \wedge w_2)$ is the probability that w_1 and w_2 occur together and $p(w_1)$ and $p(w_2)$ are the occurrence probabilities of separate words. The ratio between $p(w_1 \wedge w_2)$ and $p(w_1) \times p(w_2)$ is thus a measure of the degree of statistical dependence between the words. $PMI(w_1, w_2) = 0$ means that the words are independent, positive values indicate dependency between the words (the words often occur together). The final sentiment orientation SO of the phrase is computed as follows:

$$SO(phrase) = PMI(phrase, \text{“excellent”}) - PMI(phrase, \text{“poor”}) \tag{5.8}$$

Other unsupervised approaches can be found in [Xianghua et al., 2013, García-Pablos et al., 2018]. Liu [2012] considers the lexicon-based (see Section 5.1) methods as another type of unsupervised learning but, we treat it as a separate approach.

5.4 Hybrid Approaches

The combination of traditional machine learning (supervised or unsupervised) and lexicon-based approach is called the *hybrid approach* [Medhat et al., 2014, Giachanou and Crestani, 2016, Singh et al., 2018].

Zhang et al. [2011] proposed a hybrid approach for entity-level SA on Tweets. They firstly employed a lexicon-based approach to perform entity-level SA with high precision but low recall. In order to improve the recall of the method, they automatically identified additional tweets. They then trained the Support Vector Machines classifier on Tweets that were labeled automatically by the lexicon-based approach. Zhang et al. [2015a] classify sentiment polarity of chines comments on clothing products. They used word embeddings to cluster similar features in the selected domain. Then, the lexicon-based and part-of-speech based feature selection methods are employed to extract features. Using the generated features, the SVM classifier is trained.

Another related work can be found in [Feldman et al., 2011, Khuc et al., 2012, Mudinas et al., 2012, Khan et al., 2014, Ortigosa et al., 2014, Kolchyna et al., 2015, Khan et al., 2016].

5.5 State-of-the-Art of Sentiment Analysis

The latest state-of-the-art methods in SA are based on neural networks, deep learning techniques and contextualized representations of words (contextualized word vectors). The initial deep learning models for used traditional static word embeddings like word2vec [Mikolov et al., 2013a], GloVe [Pennington et al., 2014] or fastText [Bojanowski et al., 2017] as a sequence input, followed by next hidden layers using recurrent neural network (RNN) (e.g., [Baziotis et al., 2017]) or convolutional neural network (CNN) (e.g., [Kim, 2014]) or using their combination [Wang et al., 2016]. The RNN is implemented either with Long Short-Term Memory [Hochreiter and Schmidhuber, 1997] (LSTM) layer (eventually Bidirectional LSTM [Graves and Schmidhuber, 2005]) or with Gated Recurrent Unit [Cho et al., 2014c]. The output is then passed to a fully-connected dense layer followed by a softmax layer. The output of the softmax layer is a probability distribution over all possible output classes, see Figure 5.2 for visualisation of such example architecture with BiLSTM.

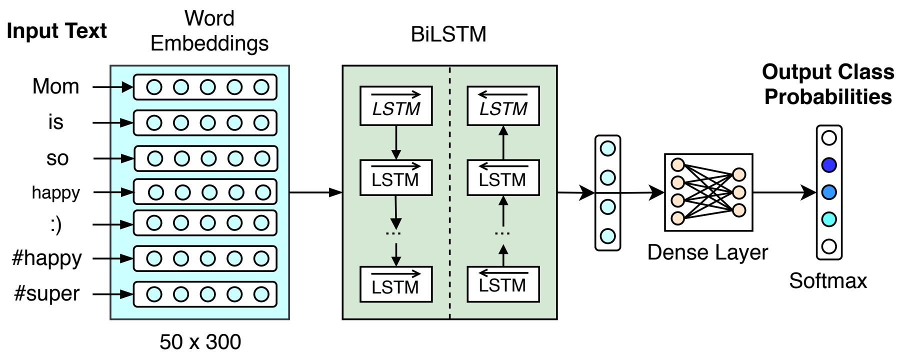


Figure 5.2: Example of neural network architecture for SA. The input text is represented by word vectors with dimension 300, the maximum length of the input sequence is 50. Then the input is passed to one layer of BiLSTM and one fully connected dense layer. The output probabilities are computed with the last softmax layer.

The most recent state-of-the-art approaches use pre-trained models with hundreds of millions of parameters, for example, *Embeddings from Language Models* (ELMo) [Peters et al., 2018], *Bidirectional Encoder Representations from Transformers* (BERT) [Devlin et al., 2019], *Universal Language Model Fine-tuning* (ULMFiT) [Howard and Ruder, 2018], *Generative Pre-Training* (GPT 2) [Radford et al., 2019] and *Generalized Autoregressive Pre-training* (XLNet) [Yang et al., 2019]. These models are trained on large corpora of unlabeled text and they are capable of producing contextualized word or sentence embeddings, see Chapter 4.

The pre-trained model is then fine-tuned on a specific downstream task. Next, we mention the most recent state-of-the-art approaches for SA and their results. They are usually built on top of the mentioned pre-trained models. We do not explain them in detail, but a brief description of some of them is provided in Section 4.2.

Bidirectional Encoder Representations from Transformers (BERT) [Devlin et al., 2019] is a recent breakthrough model for language representation based on neural network architecture called *Transformer*, see Section 4.2.3 for BERT description. In the original paper, BERT achieved 94.9% of accuracy on *SST-2* dataset. Sun et al. [2019b] utilize BERT fine-tuning methods for text classification tasks and they were able to achieve 95.79% of accuracy on *IMDb* dataset, 28.62 of error rate[3] on *Yelp-Fine* dataset and 1.81 of error rate on *Yelp-Binary* dataset.

XLNet Yang et al. [2019] is the newest BERT-like pre-trained model by Google Brain. It outperforms BERT on 20 NLP tasks, including four well-known datasets for sentiment polarity classification in which it holds the best results among any other models at all. More concretely, for polarity detection, XLNet achieved 96.21% of accuracy on *IMDb* dataset, 96.8% of accuracy on *SST-2* dataset, 27.8% of error rate on *Yelp-Fine* dataset and 1.55 of error rate on *Yelp-Binary* dataset.

In aspect-based sentiment, Sun et al. [2019a] utilized BERT specifically for aspect-based sentiment task. They constructed an auxiliary sentence from the aspect and converted the aspect-based task to a sentence-pair classification task (similar to question answering task). They obtained state-of-the-art results for *SentiHood* dataset, they achieved 87.1% of F_1 score for the aspect extraction task and 93.6% of accuracy for sentiment polarity prediction. For older deep learning approaches and for more comprehensive survey, see [Zhang et al., 2018].

[3]The error rate is computed as $1 - \textit{accuracy}$

Chapter 6

Multilingual Sentiment Analysis

In this chapter, we describe two general approaches for multilingual SA in Section 6.1. In Section 6.2, we elaborate cross-lingual word embeddings in more detail. Cross-lingual word embeddings are one of the two general approaches for multilingual SA.

6.1 Multilingual Approaches

At the very beginning of the SA research, papers were almost exclusively focused on English, but in the following years, the attention has moved and research has been made even for other languages, moreover multilingual and cross-lingual methods were developed in recent years. However, developing multilingual methods for most NLP tasks is still an open and challenging problem. Also, there is still preserving problem with English oriented datasets, in other words, there is very little (or any) of SA datasets for other languages, so-called *low-resource languages* [Liu, 2012, Balabantaray et al., 2012, Dashtipour et al., 2016, Chen et al., 2018, Ruder et al., 2019].

The *multilingual* and *cross-lingual* concepts are very closely related and there is a large overlap between them, but they are not equal. We would like to mention the difference between them, generally in NLP. The *multilingual* system or approach is a system that can process text (perform some NLP task) on more than one language. There can be part of the approach that is common for all languages (e.g., some common preprocessing steps) and language-specific, for example, training sentiment classifier for each language separately. On the other hand, the *cross-lingual* system transfers or adapts knowledge of other languages to perform the task. The approach (or its parts) for a particular language depends on approach, data or tool of the other languages.

For example, we can train a sentiment classifier for low-resource languages using English data and machine translation. A sentiment classifier is trained using the English data and any known supervised approach. When a text is needed to be classified, the text is translated into English and classified with the trained classifier. The second option is that the English data are translated into all required languages and then for each language, a single classifier is trained on the translated data and the machine translation is no longer needed. The reason for this approach can be that the machine translation tool does not have to be available when the system is deployed or it can be too slow for a production environment or it can be too expensive in case of using it as a paid service.

In this example, low-resource languages depend on English. The cross-lingual approach is usually also multilingual and applicable for all involved language (first option), but as it is evident from the second mentioned approach, the system does not have to be usable for all involved language (English) in this example. The *multilingual* and *cross-lingual* concepts are often used interchangeably, even though they are not equal.

The primary motivation for developing cross-lingual methods is to enable *transfer learning* between languages, in most cases between resource-rich language (e.g., English) and low-resource language. The goal is to develop methods that will allow us to use resources (data, methods etc.) of resource-rich languages for low-resource languages in a certain NLP task [Ruder et al., 2019]. The resource-rich language is a language that has enough available resources (any type of data or methods) for a specific NLP task. Let us explain the concept of *target* and *source* language. The *source* language denotes language used for obtaining some knowledge or training data, usually it is the resource-rich language (English in the example above). The *target* language is usually the low-resource language and the goal is to solve the task in the target language.

Next, we divide the cross-lingual approaches into two categories – *machine translation* based approaches and *cross-lingual embeddings* based approaches. They are divided according to the technique used for knowledge transfer between language. Since not all existing methods perfectly fit this categorization, we place all other methods under the *cross-lingual embeddings* based approaches.

6.1.1 Machine Translation Approaches

As we already mentioned, *machine translation* (MT) can be used as a tool for building cross-lingual methods for SA. Liu [2012] mentions that there are three main strategies:

1. Create a classifier for the source language and translate the evaluated text in the target language (low-resource, e.g., Czech) into the source language and classify it using a source language classifier.

2. Translate source training data into the target language and build a classifier in the target language.

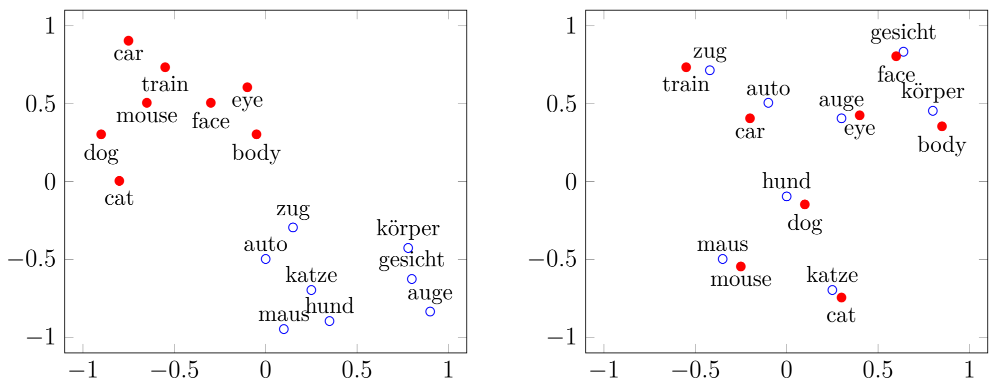
3. Translate sentiment lexicon in the source language to the target language and build a lexicon-based method in the target language.

The flawless MT system would allow achieving similar state-of-the-art results on the target language text using only the source language's training data. The current machine translation achieves very promising results [Edunov et al., 2018], yet they are still not perfect [Ruder et al., 2019]. Such MT system brings errors into the translated text and thus decreasing the performance of a classifier that uses these data. The quality of the translation very significantly affects the performance of the classifier.

The early approaches and experiments with machine translation can be found in Wan [2008, 2009] for Chinese, in Brooke et al. [2009] for Spanish, [Kim and Hovy, 2006] for German. Banea et al. [2010] experimented with sentence-level subjectivity classification in five languages. Lu et al. [2011] used labeled data for two languages and unlabeled parallel corpus to improve sentiment polarity classification in both languages. Balahur and Turchi [2012] employed three distinct machine translation systems and supervised machine learning in French, German and Spanish. In [Balahur and Turchi, 2014], authors extended the work from [Balahur and Turchi, 2012] with tf-idf weighting of unigram features. Balahur and Perea-Ortega [2015] experimented with sentiment polarity of tweets using n-gram features and sentiment lexicons along with a supervised learning approach. Singhal and Bhattacharyya [2016] translated reviews/sentences in Hindi, Marathi, Russian, Dutch, French, Spanish, Italian, German and Portuguese into English and then they used English word embeddings, polarities from a sentiment lexicon and a CNN model for sentiment classification. Zhou et al. [2016] also used a machine translation to create training data in the target low-resource language. Then they built a bilingual LSTM network with attention mechanism for sentiment polarity detection of documents. In [Can et al., 2018], the authors studied possibilities of usage of English model for SA in Russian, Spanish, Turkish and Dutch, where the annotated data are more limited.

6.1.2 Sentiment Analysis with Cross-Lingual Embeddings

Word embeddings (WE) allow us to capture the meaning of words in a vector representation and in recent years they turned out to be extremely useful and important in building NLP systems. Cross-lingual word embeddings (CWE) project monolingual embeddings spaces into one shared space where vectors in different languages for semantically close words are similar, as shown in Figure 6.1. Methods for cross-lingual projection are described in Section 6.2.



(a) Embeddings before the projection. (b) Embeddings after the projection.

Figure 6.1: Sample visualisation of monolingual embeddings for English and German before and after their projections into a shared cross-lingual space.

In SA (and in NLP generally), CWE allows transferring of knowledge between languages, which is very useful especially, for low-resource languages. For example, with CWE, approaches using monolingual embeddings that were already developed can be trained with the training data from resource-rich language and CWE. Thanks to the cross-lingual embeddings and its properties, samples from the low-resource language can now be predicted with the trained model.

Further, we mention methods based on CWE or approaches that are similar or related to the idea of transforming knowledge between languages without machine translation.

Jain and Batra [2015] employed recursive autoencoder architecture and sentence aligned corpora of English and Hindi texts to create a system for cross-lingual sentiment polarity classification. The model was evaluated on a Hindi movie reviews dataset. Zhou et al. [2015] proposed a method for learning bilingual sentiment word embeddings specifically for cross-lingual sentiment polarity classification. Then, they trained SVM algorithm using the embeddings for sentiment polarity classification. Barnes et al. [2016] compared multiple techniques for aspect-based cross-lingual sentiment classification.

Abdalla and Hirst [2017] experimented with linear transformation method from [Mikolov et al., 2013b] on English, Spanish and Chinese. In [Barnes et al., 2018], authors presented a method for training bilingual sentiment word embeddings which are jointly optimized to represent (a) semantic information in the source and target languages, that are bound to each other through a small bilingual dictionary and (b) sentiment information, which is annotated on the source language only. They showed the effectiveness of their approach by comparing it with other cross-lingual methods. Dong and

De Melo [2018] proposed an algorithm for cross-lingual SA with CNN that they evaluated on nine different languages.

Chen et al. [2018] trained adversial neural network that uses bilingual embeddings to train cross-lingual system for polarity classification in Chinese and Arabic using only English train data. They outperformed other two state-of-the-art systems.

Recently a multilingual language model *XLM-R* was proposed by [Conneau et al., 2019]. The model was evaluated using multiple NLP cross-lingual tasks and achieved new state-of-the-art results on many of them. The model was also evaluated on the GLUE [Wang et al., 2019] benchmark that contains an English dataset for binary sentiment classification task SST-2 [Socher et al., 2013]. The model achieved very competitive results (95% of accuracy) compared with other state-of-the-art monolingual systems.

6.2 Cross-Lingual Word Embeddings

Introducing static word embeddings like word2vec, GloVe or fastText brought a huge performance boost in almost all NLP tasks. In recent years, these monolingual embeddings are being substituted by more powerful contextualized vector representations (e.g., BERT, ELMo, XLNet etc.). Nowadays, static WE are used in experiments that aim to create *cross-lingual* word embeddings (CWE). CWE allows representing words multilingually. It means that vectors representing semantically close words in different languages are similar, see Figure 6.1.

Cross-lingual methods are being developed mainly for two reasons. (1) they enable us *to compare the meaning of words across languages*, which is key to machine translation or cross-lingual information retrieval. (2) CWE allows transferring knowledge between languages, in most cases between resource-rich and low-resource languages [Ruder et al., 2019].

The most recent works like multilingual BERT (mBERT) [Devlin et al., 2019] or XLM-R [Conneau et al., 2019] focus on multilinguality in using contextualized word vectors, but most of the research so far has been done with static word embeddings. Next, we will mention approaches that are usually focused on mapping only two monolingual embeddings. These CWE are called bilingual word embeddings (BWE). The final (future) and a much more difficult goal of CWE is to learn a shared embedding space between words in all languages [Ruder et al., 2019].

Ruder et al. [2019] categorize cross-lingual embedding methods mainly by the parallel data required by the methods. The parallel data represents the bilingual supervision signal that allows us to learn to align two monolingual spaces into one cross-lingual space. The main differences between the models usually come from the required data. The other differences are not so important since they are just implementation details for the specific architecture. To support this claim, they showed that the methods usually optimize still the same or very similar learning objective (it is just written in different forms).

The parallel data used by the methods have two key properties that distinguish them: (1) *type of alignment* and (2) *comparability* of the parallel data. The parallel data alignment defines whether the data are aligned at the level of words, sentences or documents. The comparability means how much are the parallel data similar, i.e., whether it is a literal translation (data are *parallel*) or whether the parallel data are just similar (*comparable*). Here, we do not distinguish between them. Finally, we can define the basic categorization according to type of data alignment:

1. **Word-level alignment**: Most methods use data aligned at word-level as bilingual or multilingual dictionaries of translated pairs of words. Such dictionaries are easy to obtain for most languages.The majority of these approaches use pre-trained monolingual word embeddings, a bilingual dictionary and linear transformation.

2. **Sentence-level alignment**: A parallel corpus aligned at a sentence-level is another type of data used by cross-lingual methods. An example of commonly used sentence-level aligned dataset is Europarl corpus [Koehn, 2005], which usually used for training machine translation systems.

3. **Document-level alignment**: A parallel corpus that contains translated documents in different languages. An example of such a corpus is Wikipedia, where many pages (about the same topic) are in multiple languages (topic aligned corpus).

Next, we will focus on methods that use data aligned at a word-level, concretely on *mapping-based approaches*, since these methods are very common and easy to obtain approaches for CWE. We summarize the multi-level categorization of methods for CWE in Figure 6.2. We do not describe all mentioned types of methods, but we provide a brief description and examples of some of them as they were described in [Ruder et al., 2019]. The detailed description of the other not mentioned methods, including the original papers, can be found in the same paper.

6.2.1 Methods for Word-Level Alignment Data

Mapping-based approaches usually transform pre-trained monolingual word embeddings using linear transformation and bilingual dictionaries into one joint space [Brychcín, 2020]. Linear transformation allows transforma-

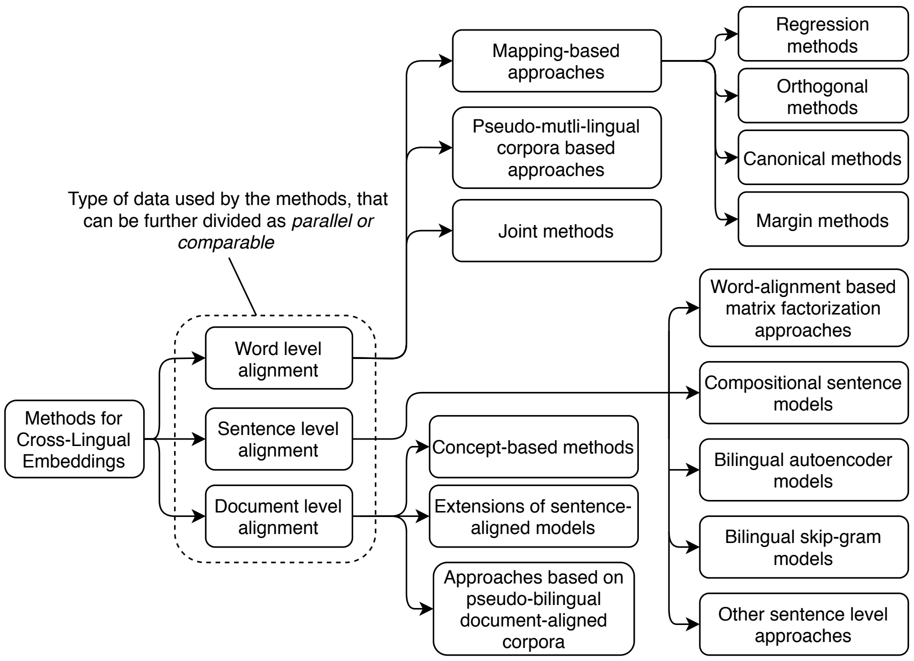

Methods for Cross-Lingual Embeddings
Type of data used by the methods, that can be further divided as *parallel or comparable*
Word level alignment
Sentence level alignment
Document level alignment
Mapping-based approaches
Pseudo-mutli-lingual corpora based approaches
Joint methods
Regression methods
Orthogonal methods
Canonical methods
Margin methods
Word-alignment based matrix factorization approaches
Compositional sentence models
Bilingual autoencoder models
Bilingual skip-gram models
Other sentence level approaches
Concept-based methods
Extensions of sentence-aligned models
Approaches based on pseudo-bilingual document-aligned corpora

Figure 6.2: An overview of categorization of methods for creating CWE according to [Ruder et al., 2019].

tion between two vector spaces[1] using affine transformations, e.g., scaling, rotation, translation, reflection.

Let us define monolingual vector spaces and dictionary of translated pairs of words. The dictionary D contains n translated pairs of words (called *seed words* or *seed dictionary*) $((w_1^s, w_1^t), (w_2^s, w_2^t), \ldots, (w_n^s, w_n^t))$. Vector space of the source language s is represented by a matrix $\mathbf{X}^s \in \mathbb{R}^{n \times d}$ and vector space of the target language t is represented by a matrix $\mathbf{X}^t \in \mathbb{R}^{n \times d}$ where n is the size of the seed dictionary and d is a dimension of the vector spaces. Each word w_i from the dictionary D is in matrices $\mathbf{X}^s$ and $\mathbf{X}^t$ represented by vectors $\mathbf{x}_i^s$, $\mathbf{x}_i^t$, respectively.

Then, the linear transformation transforms the vector space $\mathbf{X}^s$ of the source language into the vector space $\mathbf{X}^t$ of the target language using transformation matrix $\mathbf{W}^{s \to t} \in \mathbb{R}^{d \times d}$ by the following matrix multiplication:

$$\widehat{\mathbf{X}}^s = \mathbf{W}^{s \to t}\mathbf{X}^s \tag{6.1}$$

where $\widehat{\mathbf{X}}^s$ is the transformed source vector space in the target space.

The goal is to estimate the transformation matrix $\mathbf{W}^{s \to t}$. The simplest methods are called *regression methods*. They map the vector space of source language into the vector space of the target language by maximizing the

[1]By space we mean word embeddings also called semantic space, expressed by matrix $\mathbf{X}$.

similarity between the transformed source matrix and the original matrix of the target space. The first of these methods was proposed by [Mikolov et al., 2013b]. They estimate $\mathbf{W}^{s\to t}$ with stochastic gradient descent by minimizing the square Euclidean distance, i.e., mean squared error (MSE), between the pairs of vectors $(\mathbf{x}_i^s, \mathbf{x}_i^t)$ of words from the seed dictionary after applying the transformation, thus the goal is to minimize MSE computed as follows:

$$MSE = \sum_{i=1}^{n} \left\| \mathbf{W}^{s\to t}\mathbf{x}_i^s - \mathbf{x}_i^t \right\|^2 \tag{6.2}$$

It can also be rewritten using only matrices and Frobenius norm as follows:

$$MSE = \left\| \mathbf{W}^{s\to t}\mathbf{X}^s - \mathbf{X}^t \right\|_F^2 \tag{6.3}$$

Orthogonal methods constraint the transformation matrix $\mathbf{W}^{s\to t}$ to be *orthogonal* in order to improve the regression method proposed by [Mikolov et al., 2013b]. Matrix $\mathbf{W}$ is orthogonal when it is a square matrix and the columns and rows are orthonormal vectors ($\mathbf{W}^\intercal\mathbf{W} = \mathbf{W}\mathbf{W}^\intercal = I$, where I is the identity matrix). The optimal transformation matrix $\mathbf{W}^{s\to t}$ is the given by:

$$\mathbf{W}^{s\to t} = \mathbf{V}\mathbf{U}^\intercal \tag{6.4}$$

where matrices $\mathbf{V}$ and $\mathbf{U}$ are computed by *Singular Value Decomposition* (SVD) [Golub and Reinsch, 1970] of $\mathbf{X}^{t^\intercal}\mathbf{X}^s = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\intercal$ which is derived in [Artetxe et al., 2016].

Canonical methods are based on a method called *Canonical Correlation Analysis* (CCA), which is a way of measuring a linear relationship between two multivariate variables (i.e., vectors) [Brychcín, 2020]. These methods map both monolingual vector spaces $\mathbf{X}^s$ and $\mathbf{X}^t$ to a different shared space represented by matrix $\mathbf{Y}^o$, thus canonical methods compute two transformation matrices $\mathbf{W}^{s\to o}$ for the source language and $\mathbf{W}^{t\to o}$ for the target language, in order to map their vector spaces into one shared space $\mathbf{Y}^o$. The transformation matrices are computed by minimizing the negative correlation (denoted as $Ncor$) between the vectors $\mathbf{x}_i^s$ (source) and $\mathbf{x}_i^t$ (target) projected into the shared space $\mathbf{Y}^o$, given by:

$$Ncor = -\sum_{i=1}^{n} \rho(\mathbf{W}^{s\to o}\mathbf{x}_i^s, \mathbf{W}^{t\to o}\mathbf{x}_i^t) \tag{6.5}$$

The correlation $\rho(\mathbf{W}^{s\to o}\mathbf{x}_i^s, \mathbf{W}^{t\to o}\mathbf{x}_i^t)$ is computed as follows:

$$\rho(\mathbf{W}^{s\to o}\mathbf{x}_i^s, \mathbf{W}^{t\to o}\mathbf{x}_i^t) = \frac{cov(\mathbf{W}^{s\to o}\mathbf{x}_i^s, \mathbf{W}^{t\to o}\mathbf{x}_i^t)}{\sqrt{var(\mathbf{W}^{s\to o}\mathbf{x}_i^s) \times var(\mathbf{W}^{t\to o}\mathbf{x}_i^t)}} \tag{6.6}$$

where cov the covariance and var is the variance.

Faruqui and Dyer [2014] used the method as first for the mapping two monolingual word embeddings space into one cross-lingual space. Ammar et al. [2016] extended the approach for multiple languages. Using the approach from [Ammar et al., 2016] the transformation matrix $\mathbf{W}^{s\to t}$ can be computed as follows:

$$\mathbf{W}^{s\to t} = \mathbf{W}^{s\to o}(\mathbf{W}^{t\to o})^{-1} \tag{6.7}$$

Margin methods use different loss functions to optimize the transformation matrix. Lazaridou et al. [2015] used a *max-margin based ranking loss* (MML) instead of mean square error (MSE). They used MML in order to reduce *hubness* and the idea is to rank the correct translations of word w_i (i.e., vectors $\mathbf{x}_i^s$ and $\mathbf{x}_i^t$) higher than random translation (negative example) of word w_i (i.e., vectors $\mathbf{x}_i^s$ and $\mathbf{x}_j^t$) which is given by minimizing the following function:

$$MML = \sum_{i=1}^{n}\sum_{j\neq i}^{k} \max\{0, \gamma - \cos(\mathbf{W}^{s\to t}\mathbf{x}_i^s, \mathbf{x}_i^t) + \cos(\mathbf{W}^{s\to t}\mathbf{x}_i^s, \mathbf{x}_j^t)\} \tag{6.8}$$

where γ and k are tunable hyper-parameters denoting the margin and the number of negative examples, respectively. The negative example $\mathbf{x}_j^t$ is a vector for random word that is not the translation of word w_i. In other words, the goal of the optimization is to estimate the matrix $\mathbf{W}^{s\to t}$ in that way that $\cos(\mathbf{W}^{s\to t}\mathbf{x}_i^s, \mathbf{x}_i^t)$ (i.e., translated word pairs) is highest as possible and $\cos(\mathbf{W}^{s\to t}\mathbf{x}_i^s, \mathbf{x}_j^t)$ (i.e., random word pairs) is as smallest as possible.

Methods using pseudo-multilingual corpora are another type of approach that uses word-level alignment data. They are trained on a modified corpus. Randomly chosen words from the corpus in the source language are replaced with their translation. The replaced words are part of the seed dictionary. Xiao and Guo [2014] applied this approach first. They first create a bilingual dictionary and then map each pair of words from the dictionary to the same vector in the vector space. Next, they train word embeddings using the same approach as [Collobert and Weston, 2008] with sentences in target and source language and using max-margin based ranking loss. Another approach using pseudo-multilingual corpora can be found in [Gouws and Søgaard, 2015, Ammar et al., 2016, Duong et al., 2016, Adams et al., 2017]

Joint methods differ from the previously mentioned method in that way that they optimize monolingual and cross-lingual objectives at the same time jointly. Their description, along with concrete examples of models, can be found in [Ruder et al., 2019].

6.2.2 Methods for Sentence-Level Aligned Data

Research in machine translation brought a sufficient amount of sentence-aligned data for some European languages, but generally, obtaining such data is much more difficult and expensive than word-level aligned data. Methods using sentence-level alignment data can be categorized into four sub-categories: (1) *word-alignment based matrix factorization approaches*, (2) *compositional sentence models*, (3) *bilingual autoencoder models*, (4) *bilingual skip-gram models* and (5) *other sentence-level approaches* [Ruder et al., 2019].

Word-alignment based matrix factorization approaches use matrix factorization methods and they also usually require some data to be aligned at the word-level. The common approach is to assemble an alignment matrix $\mathbf{A}^{s\to t}$ between source and target language. The alignment matrix can be obtained in an unsupervised manner and contain information how many times was each word from source language aligned (translated) to word in the target language. More precisely, $\mathbf{A}^{s\to t}_{ij}$ denotes how many times was the i-th in the target language aligned to j-th word in the source language. Then, the assumption is that if a word the in source language is aligned with more than one word in the target language than the word holds multiple meanings and its representation should be composed of its aligned words.

Zou et al. [2013] applied this approach for creating English-Chinese bilingual word embeddings. They first, trained monolingual embeddings and they constructed two alignment matrices $\mathbf{A}^{s\to t}$ and $\mathbf{A}^{t\to s}$ (one for each direction of translations). Next, they jointly minimize the following objective functions $J_{s\to t}$ and $J_{t\to s}$ in both directions:

$$J_{s\to t} = \left\|\mathbf{X}^t - \mathbf{A}^{s\to t}\mathbf{X}^s\right\|^2 \tag{6.9}$$

$$J_{t\to s} = \left\|\mathbf{X}^s - \mathbf{A}^{t\to s}\mathbf{X}^t\right\|^2 \tag{6.10}$$

The objective functions are similar to the MSE approach from [Mikolov et al., 2013b], but the alignment matrices $\mathbf{A}^{s\to t}$ and $\mathbf{A}^{t\to s}$ are fixed (they do not change during training) and the embeddings matrices $\mathbf{X}^s$ and $\mathbf{X}^t$ are optimized during training. Another related work can be found in [Shi et al., 2015, Huang et al., 2015, Vyas and Carpuat, 2016, Guo et al., 2015]

Compositional sentence models learn cross-lingual embeddings based on the aligned sentence representations. In other words, the model optimizes the vector representation of each sentence (in target and source language) to be close to each other. Hermann and Blunsom [2013] represented each sentence as a sum of embeddings vectors of the sentence words. More precisely, the representation $\mathbf{y}^s$ of the sentence $sent^s$ in the source language s is computed as follows:

$$\mathbf{y}^s = \sum_{i=1}^{|sent^s|} \mathbf{x}_i^s \tag{6.11}$$

where $\mathbf{x}_i^s$ is a vector representing word w_i. The representation $\mathbf{y}^t$ of sentence $sent^t$ in the target language t is computed in the same way. Then they utilize the vector representation with the following objective function J_{MML}:

$$J_{MML} = \sum_{(sent^s, sent^t)}^{C} \sum_{i=1}^{k} \max(0, 1 + E_{dist}(sent^s, sent^t) - E_{dist}(sent^s, sent_i^t)) \tag{6.12}$$

where k is a number of negative random non-aligned sentences, C is a set of aligned sentences. The distance $E_{dist}(sent^s, sent^t)$ between sentence representations is computed as follows:

$$E_{dist}(sent^s, sent^t) = ||\mathbf{y}^s - \mathbf{y}^t||^2 \tag{6.13}$$

The optimization is based on max-margin loss and it pushes the representation of aligned sentences to be closer than the negative random non-aligned sentences.

Bilingual encoder encodes the representation of the sentence in the source language into a hidden state and then reconstructs the representation of the aligned sentence in the target language. Such approach was applied in [Lauly et al., 2014, Chandar et al., 2014].

Bilingual skip-gram methods are the cross-lingual extensions of the original skip-gram with negative sampling (SGNS) algorithm. These methods a) jointly optimize the objectives for each language and b) optimize one shared cross-lingual objective for more details, see [Ruder et al., 2019].

Other sentence-level approaches have been a subject of research in recent years, especially with the rise of contextualized word vectors techniques based on pre-trained language models that became very popular, for example, ULMFiT [Howard and Ruder, 2018], BERT [Devlin et al., 2019], GPT-2 [Radford et al., 2019] or XLNet [Yang et al., 2019], see Section 4.2. Some of these were utilized to allow learning cross-lingual representation for multiple languages at once. Conneau and Lample [2019] introduced an unsupervised method for learning cross-lingual representations along with a variant of a supervised approach that allows improving the cross-lingual model when parallel data is available. Another completely unsupervised work is presented in [Artetxe et al., 2018][2]. Recently, a multilingual language model *XLM-R* was proposed by [Conneau et al., 2019] specifically aimed to train multilingual language models.

6.2.3 Methods for Document-Level Aligned Data

Methods using data aligned on document-level are not so common as the two previous two categories, but some research has been done. These methods

[2]According to the categorization, this method belongs under the *mapping-based approaches*

mostly rely on comparable documents (not parallel, i.e., exact translations). The natural source of such documents is Wikipedia, where many topics are covered by multiple languages. These methods can be further divided into three types and some of them are based on principles from the previous two categories. (1) *Approaches based on pseudo-bilingual document-aligned corpora* modify existing documents by randomly replacing words with their translations (same idea as in case of word-level aligned data). Vulic and Moens [2016] experimented with this type of data. The last two categories are (2) *concept-based methods* and (3) *extensions of sentence alignment methods* that extend methods for sentence aligned data in that way that they are applicable to comparable data [Ruder et al., 2019].

6.2.4 Hubness

Hubness [Radovanović et al., 2010] is a feature that occurs in high-dimensional spaces (and cross-lingual embeddings spaces are usually high-dimensional). In the space, there are *hubs*, i.e., points (in our case word vectors) that are nearest neighbours, of many other points (other word vectors), but they should not be similar[3] to each other because their semantic or meaning is very different. Hubness negatively affects the quality of cross-lingual spaces.

6.2.5 Evaluation

With the rise of cross-lingual methods, their evaluation became an important part of their development. There are two main types of evaluation tasks: *intrinsic* and *extrinsic*, that measure the quality of the embeddings.

In the extrinsic evaluation, the cross-lingual embeddings are used in downstream tasks (real-world problems where cross-lingual embeddings can be applied) and the quality of the embeddings is based on the results in the specific downstream task.

The intrinsic evaluation is a specially designed task for the evaluation of cross-lingual embeddings testing their ability to capture semantic or syntactic relationships between words in comparison with human judgments. The major disadvantage of intrinsic tasks is that if certain cross-lingual embeddings perform well on them, it does not necessarily imply a good performance on downstream tasks [Ruder et al., 2019]. One of the most common intrinsic tasks is called *bilingual lexicon induction*. For source language s and given list of N source language words $w_1^s, w_2^s, \ldots, w_N^s$ the goal is to find for each word w_i^s its best-suited translation represented by a word w_i^t in target language t. The common approach is to find word w_i^t that is the nearest neighbor (the most similar[3] vector) to the word w_i^s in the cross-lingual semantic space.

[3]The similarity can be measured, for example, with cosine distance.

Chapter 7

Summary

In the previous chapters, we presented the theory about opinion mining stated in [Liu, 2012]. Then, we explored and described SA tasks and their related approaches and data resources. We described recent state-of-the-art works and techniques (mostly demonstrated in English). At the end, we discussed multilinguality in SA and methods to deal with the lack of data in low-resource languages. From the described papers, we can see that in recent years the neural networks and deep learning techniques are used in most cases but for some situations in practice (usually when there is no or small amount of annotated data) older machine learning algorithms and approaches can be used as well. Despite the great improvement in recent years, especially in English, there are still open problems and challenges in SA. We observe that there are two main open challenges:

- Domain adaption.
- Sentiment analysis in low-resource languages.

In the case of *domain adaption*, systems are usually trained using supervised machine learning algorithms that require annotated data and these data usually come from one domain (e.g., movie reviews). The system performs well on data from this domain but it usually drops when the system is used on different domain. This problem can be tackled by introducing methods that allow the system to adapt to the new domain.

The lack of data in low-resource language can be solved in two ways. (1) Annotating or creating data for low-resource languages. For some tasks, the data can by obtained automatically or semi-automatically, (e.g., *distant supervision*) but data for other tasks have to be annotated manually which is very expensive and time-consuming. (2) Applying or developing new cross-lingual techniques that allow transfer knowledge from resource-rich languages to low-resource languages.

Chapter 8

Preliminary Results and Future Work

In this chapter, I summarize my work and preliminary results. Next, I propose future work in a field of SA and related tasks. Lastly, the aims of the doctoral thesis are specified.

8.1 Challenges and Future Work

As it was already mentioned in summary in Section 7, regardless of the improvement in SA, there are still open problems. I identified two main challenges: (1) domain adaption and (2) sentiment analysis in low-resource languages.

The task of *domain adaption* is to adapt system or approach to be usable for data from a domain other than the one used for the development of the system. Nowadays, a big part of datasets for SA comes from a narrow set of domains, e.g., movie reviews or social media websites like Twitter. Thanks to insufficient data for some domains, train systems for such domains is difficult. The challenge is to develop novel methods and approaches that could solve this task.

Despite the improvement in SA, there is still a lack of data for low-resource languages or generally other languages than English. I see two possible ways to tackle this problem: (1) create new datasets and resources for languages other than English or (2) apply cross-lingual techniques that allow the transfer of knowledge between languages.

A new dataset can be created either by a manual annotation or by developing methods that can obtain labeled datasets automatically or semi-automatically (e.g., *distant supervision*). Unfortunately, some tasks are too difficult to be labeled automatically and the annotation must be done manually, which is a very time-consuming and expensive process. In the case of these tasks, the cross-lingual techniques can be applied with existing datasets from a resource-rich language (e.g., English) to transfer the knowledge to a low-resource language. This approach allows to perform the task of SA on languages with little or even no training data. In the optimal case, the same approach can be potentially used for performance improvement, even for languages with a sufficient amount of data. The system is then trained on data from both languages, which could lead to the performance improvement thanks to extended training data.

In my future work, I would like to perform SA and other related tasks in languages other than English, including low-resource languages. I want to introduce new datasets that allow to use traditional machine learning approaches for SA. Next, I want to use cross-lingual techniques and data from resource-rich languages and apply them to SA (and other related tasks) and thus enable performing SA in other languages. Since most current approaches use neural networks and deep learning techniques, I also plan to use these techniques in my future work.

The very recent generalized language models like BERT or GPT 2 (see Section 4.2) seem to be very powerful tools for any NLP task in English. I would like to investigate their possibilities to utilize them for other languages than English in SA.

8.2 Preliminary Results

In our initial work [Přibáň et al., 2018], we developed a system that can detect an intensity of a given emotion in English, Spanish and Arabic tweets. Given a tweet and one of four emotions (anger, fear, joy or sadness), the output of the system is a degree of the intensity of the given emotion, where the degree is either a real value number (regression task) or one of four classes (classification task) corresponding to the strength of the intensity.

In [Přibáň and Martínek, 2018], we employed a neural network with BiLSTM to detect one of six implicit emotion in a given tweet, i.e., emotions that are not explicitly mentioned in the text. We achieved 0.657 of F_1 macro score.

In another paper[1], we thoroughly evaluated multilingual systems for SA and we compared their performance on a rich collection of publicly available datasets. We also performed an in-depth error analysis and we proposed a potential solution for the misclassified examples.

In [Pražák et al., 2020], we developed an approach for detecting the lexical semantic-change in English, German, Swedish and Latin, i.e., word

[1]The paper was not published yet. It was presented at the CICLing 2019 (20th International Conference on Computational Linguistics and Intelligent Text Processing) conference and it was accepted to be published in the Lecture Notes in Computer Science (LNCS). The paper is available at `http://home.zcu.cz/~pribanp/cicling/CICLing_2019.pdf`.

sense changes over time. The paper is not directly related to SA, but we applied cross-lingual techniques that are also applicable in any cross-lingual task. In addition, the proposed approach can also be used to identify domain-specific changes of word senses in comparison to general–language. Thus, it can be very beneficial information in approaches for the domain adaption task in SA.

In terms of multilinguality in NLP, in [Přibáň et al., 2019], we made available dataset for fact-checking task in Czech, Polish and Slovak. In [Piskorski et al., 2019], we present the *Second Multilingual Named Entity Challenge in Slavic languages* competition. The proposed task is to recognize mentions of named entities in news articles, their normalization and cross-lingual linking in Czech, Polish, Russian and Bulgarian. The work also contains publicly available multilingual dataset for the mentioned languages.

8.3 Aims of the Doctoral Thesis

As I already mentioned, the doctoral thesis, generally speaking, aims to perform sentiment analysis and/or other related tasks in other languages than English, either by introducing new datasets or by using cross-lingual techniques for knowledge transfer between languages. The aims of the doctoral thesis are summarized as follows:

- Tackle the problem of lack of data in languages other than English by introducing new resources.
- Perform sentiment analysis and/or related tasks in languages other than English by applying cross-lingual methods and transforming knowledge between resource-rich and other languages.
- Apply recent state-of-the-art approaches for semantic text representation to sentiment analysis and other related tasks to data other than English.

Bibliography

Mohamed Abdalla and Graeme Hirst. Cross-lingual sentiment analysis without (good) translation. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 506–515, Taipei, Taiwan, November 2017. Asian Federation of Natural Language Processing. URL https://www.aclweb.org/anthology/I17-1051.

Muhammad Abdul-Mageed and Lyle Ungar. EmoNet: Fine-grained emotion detection with gated recurrent neural networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 718–728, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-1067. URL https://www.aclweb.org/anthology/P17-1067.

Oliver Adams, Adam Makarucha, Graham Neubig, Steven Bird, and Trevor Cohn. Cross-lingual word embeddings for low-resource language modeling. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 937–947, Valencia, Spain, April 2017. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/E17-1088.

Ashutosh Adhikari, Achyudh Ram, Raphael Tang, and Jimmy Lin. Docbert: BERT for document classification. *CoRR*, abs/1904.08398, 2019. URL http://arxiv.org/abs/1904.08398.

Eneko Agirre, Lluís Màrquez, and Richard Wicentowski, editors. *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, Prague, Czech Republic, June 2007. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/S07-1000.

Parag Agrawal and Anshuman Suri. NELEC at SemEval-2019 task 3: Think twice before going deep. In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 266–271, Minneapolis, Minnesota, USA, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/S19-2045. URL https://www.aclweb.org/anthology/S19-2045.

Waleed Ammar, George Mulcaire, Yulia Tsvetkov, Guillaume Lample, Chris Dyer, and Noah A. Smith. Massively multilingual word embeddings, 2016.

Marianna Apidianaki, Saif M. Mohammad, Jonathan May, Ekaterina Shutova, Steven Bethard, and Marine Carpuat, editors. *Proceedings of The 12th International Workshop on Semantic Evaluation*, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/S18-1. URL https://www.aclweb.org/anthology/S18-1000.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. Learning principled bilingual mappings of word embeddings while preserving monolingual invariance. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2289–2294, Austin, Texas, November 2016. Association for Computational Linguistics. doi: 10.18653/v1/D16-1250. URL https://www.aclweb.org/anthology/D16-1250.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. A robust self-learning method for fully unsupervised cross-lingual mappings of word embeddings. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 789–798, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-1073. URL https://www.aclweb.org/anthology/P18-1073.

Hakob Avetisyan, Ondřej Bruna, and Jan Holub. Overview of existing algorithms for emotion classification. uncertainties in evaluations of accuracies. In *Journal of Physics: Conference Series*, volume 772, page 012039. IOP Publishing, 2016.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization, 2016.

Stefano Baccianella, Andrea Esuli, and Fabrizio Sebastiani. Sentiwordnet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining. In Nicoletta Calzolari, Khalid Choukri, Bente Maegaard, Joseph Mariani, Jan Odijk, Stelios Piperidis, Mike Rosner, and Daniel Tapias, editors, *Proceedings of the International Conference on Language Resources and Evaluation, LREC 2010, 17-23 May 2010, Valletta, Malta*. European Language Resources Association, 2010. ISBN 2-9517408-6-7. URL http://www.lrec-conf.org/proceedings/lrec2010/summaries/769.html.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1409.0473.

Rakesh C Balabantaray, Mudasir Mohammad, and Nibha Sharma. Multi-class twitter emotion classification: A new approach. *International Journal of Applied Information Systems*, 4(1):48–53, 2012.

Alexandra Balahur and José M. Perea-Ortega. Sentiment analysis system adaptation for multilingual processing: The case of tweets. *Information Processing & Management*, 51(4):547 – 556, 2015. ISSN 0306-4573. doi: https://doi.org/10.1016/j.ipm.2014.10.004. URL http://www.sciencedirect.com/science/article/pii/S0306457314000934.

Alexandra Balahur and Marco Turchi. Multilingual sentiment analysis using machine translation? In *Proceedings of the 3rd Workshop in Computational Approaches to Subjectivity and Sentiment Analysis*, pages 52–60, Jeju, Korea, July 2012. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/W12-3709.

Alexandra Balahur and Marco Turchi. Comparative experiments using supervised learning and machine translation for multilingual sentiment analysis. *Computer Speech & Language*, 28(1):56 – 75, 2014. ISSN 0885-2308. doi: https://doi.org/10.1016/j.csl.2013.03.004. URL http://www.sciencedirect.com/science/article/pii/S088523081300020X.

Alexandra Balahur, Ralf Steinberger, Erik van der Goot, Bruno Pouliquen, and Mijail Kabadjov. Opinion mining on newspaper quotations. In *Proceedings of the 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology - Volume 03*, WI-IAT '09, page 523–526, USA, 2009. IEEE Computer Society. ISBN 9780769538013. doi: 10.1109/WI-IAT.2009.340. URL https://doi.org/10.1109/WI-IAT.2009.340.

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). URL http://www.lrec-conf.org/proceedings/lrec2010/pdf/909_Paper.pdf.

Alexandra Balahur, Jesus M Hermida, and Andres Montoyo. Building and exploiting emotinet, a knowledge base for emotion detection based on the appraisal theory model. *IEEE Transactions on Affective Computing*, 3(1):88–101, 2011.

Alexandra Balahur, Jesús M Hermida, and Andrés Montoyo. Detecting implicit expressions of emotion in text: A comparative analysis. *Decision Support Systems*, 53(4):742–753, 2012.

Carmen Banea, Rada Mihalcea, and Janyce Wiebe. Multilingual subjectivity: Are more languages better? In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 28–36, Beijing, China, August 2010. Coling 2010 Organizing Committee. URL https://www.aclweb.org/anthology/C10-1004.

Francesco Barbieri, Valerio Basile, Danilo Croce, Malvina Nissim, Nicole Novielli, and Viviana Patti. Overview of the evalita 2016 sentiment polarity classification task. In Pierpaolo Basile, Anna Corazza, Francesco Cutugno, Simonetta Montemagni, Malvina Nissim, Viviana Patti, Giovanni Semeraro, and Rachele Sprugnoli, editors, *Proceedings of Third Italian Conference on Computational Linguistics (CLiC-it 2016) & Fifth Evaluation Campaign of Natural Language Processing and Speech Tools for Italian. Final Workshop (EVALITA 2016), Napoli, Italy, December 5-7, 2016*, volume 1749 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2016. URL http://ceur-ws.org/Vol-1749/paper_026.pdf.

Luciano Barbosa and Junlan Feng. Robust sentiment detection on twitter from biased and noisy data. In *Coling 2010: Posters*, pages 36–44, Beijing, China, August 2010. Coling 2010 Organizing Committee. URL https://www.aclweb.org/anthology/C10-2005.

Jeremy Barnes, Patrik Lambert, and Toni Badia. Exploring distributional representations and machine translation for aspect-based cross-lingual sentiment classification. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1613–1623, Osaka, Japan, December 2016. The COLING 2016 Organizing Committee. URL https://www.aclweb.org/anthology/C16-1152.

Jeremy Barnes, Roman Klinger, and Sabine Schulte im Walde. Bilingual sentiment embeddings: Joint projection of sentiment across languages. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2483–2493, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-1231. URL https://www.aclweb.org/anthology/P18-1231.

Christos Baziotis, Nikos Pelekis, and Christos Doulkeridis. DataStories at SemEval-2017 task 4: Deep LSTM with attention for message-level and topic-based sentiment analysis. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 747–754, Vancouver, Canada, August 2017. Association for Computational Linguistics. doi: 10.18653/v1/S17-2126. URL https://www.aclweb.org/anthology/S17-2126.

Christos Baziotis, Athanasiou Nikolaos, Alexandra Chronopoulou, Athanasia Kolovou, Georgios Paraskevopoulos, Nikolaos Ellinas, Shrikanth Narayanan, and Alexandros Potamianos. NTUA-SLP at SemEval-2018 task 1: Predicting affective content in tweets with deep attentive RNNs and transfer learning. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 245–255, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/S18-1037. URL https://www.aclweb.org/anthology/S18-1037.

Iz Beltagy, Kyle Lo, and Arman Cohan. SciBERT: A pretrained language model for scientific text. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3615–3620, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1371. URL https://www.aclweb.org/anthology/D19-1371.

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

Sasha Blair-Goldensohn, Kerry Hannan, Ryan McDonald, Tyler Neylon, George Reis, and Jeff Reynar. Building a sentiment summarizer for local service reviews. In *WWW Workshop on NLP Challenges in the Information Explosion Era (NLPIX)*, 2008.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3(null):993–1022, March 2003. ISSN 1532-4435.

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. doi: 10.1162/tacl_a_00051. URL https://www.aclweb.org/anthology/Q17-1010.

Johan Bollen, Huina Mao, and Xiaojun Zeng. Twitter mood predicts the stock market. *Journal of computational science*, 2(1):1–8, 2011.

Laura-Ana-Maria Bostan and Roman Klinger. An analysis of annotated corpora for emotion classification in text. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2104–2119, Santa Fe, New Mexico, USA, August 2018. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/C18-1179.

Margaret M Bradley and Peter J Lang. Affective norms for english words (anew): Instruction manual and affective ratings. Technical report, Technical report C-1, the center for research in psychophysiology . . . , 1999.

Felipe Bravo-Marquez, Marcelo Mendoza, and Barbara Poblete. Meta-level sentiment models for big social data analysis. *Know.-Based Syst.*, 69(1): 86–99, October 2014. ISSN 0950-7051. doi: 10.1016/j.knosys.2014.05.016. URL https://doi.org/10.1016/j.knosys.2014.05.016.

Felipe Bravo-Marquez, Eibe Frank, Saif M. Mohammad, and Bernhard Pfahringer. Determining word-emotion associations from tweets by multi-label classification. In *2016 IEEE/WIC/ACM International Conference on Web Intelligence, WI 2016, Omaha, NE, USA, October 13-16, 2016*, pages 536–539. IEEE Computer Society, 2016. ISBN 978-1-5090-4470-2. doi: 10.1109/WI.2016.0091. URL https://doi.org/10.1109/WI.2016.0091.

Samuel Brody and Noemie Elhadad. An unsupervised aspect-sentiment model for online reviews. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 804–812, Los Angeles, California, June 2010. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/N10-1122.

Julian Brooke, Milan Tofiloski, and Maite Taboada. Cross-linguistic sentiment analysis: From English to Spanish. In *Proceedings of the International Conference RANLP-2009*, pages 50–54, Borovets, Bulgaria, September 2009. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/R09-1010.

Tomáš Brychcín. Linear transformations for cross-lingual semantic textual similarity. *Knowledge-Based Systems*, 187:104819, 2020. ISSN 0950-7051. doi: https://doi.org/10.1016/j.knosys.2019.06.027. URL http://www.sciencedirect.com/science/article/pii/S0950705119302941.

Marc Brysbaert, Michaël Stevens, Simon De Deyne, Wouter Voorspoels, and Gert Storms. Norms of age of acquisition and concreteness for 30,000 dutch words. *Acta psychologica*, 150:80–84, 2014.

Sven Buechel and Udo Hahn. Emotion analysis as a regression problem — dimensional models and their implications on emotion representation and metrical evaluation. In *Proceedings of the Twenty-Second European Conference on Artificial Intelligence*, ECAI'16, page 1114–1122, NLD, 2016. IOS Press. ISBN 9781614996712. doi: 10.3233/978-1-61499-672-9-1114. URL https://doi.org/10.3233/978-1-61499-672-9-1114.

Sven Buechel and Udo Hahn. EmoBank: Studying the impact of annotation perspective and representation format on dimensional emotion analysis. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages

578–585, Valencia, Spain, April 2017. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/E17-2092.

Ethem F. Can, Aysu Ezen-Can, and Fazli Can. Multilingual sentiment analysis: An rnn-based framework for limited data. *CoRR*, abs/1806.04511, 2018. URL http://arxiv.org/abs/1806.04511.

Lea Canales and Patricio Martínez-Barco. Emotion detection from text: A survey. In *Proceedings of the Workshop on Natural Language Processing in the 5th Information Systems Research Working Days (JISIC)*, pages 37–43, Quito, Ecuador, October 2014. Association for Computational Linguistics. doi: 10.3115/v1/W14-6905. URL https://www.aclweb.org/anthology/W14-6905.

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Brian Strope, and Ray Kurzweil. Universal sentence encoder for English. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 169–174, Brussels, Belgium, November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-2029. URL https://www.aclweb.org/anthology/D18-2029.

A P Sarath Chandar, Stanislas Lauly, Hugo Larochelle, Mitesh M Khapra, Balaraman Ravindran, Vikas Raykar, and Amrita Saha. An autoencoder approach to learning bilingual word representations. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, page 1853–1861, Cambridge, MA, USA, 2014. MIT Press.

Ankush Chatterjee, Kedhar Nath Narahari, Meghana Joshi, and Puneet Agrawal. SemEval-2019 task 3: EmoContext contextual emotion detection in text. In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 39–48, Minneapolis, Minnesota, USA, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/S19-2005. URL https://www.aclweb.org/anthology/S19-2005.

Peng Chen, Zhongqian Sun, Lidong Bing, and Wei Yang. Recurrent attention network on memory for aspect sentiment analysis. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 452–461, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/D17-1047. URL https://www.aclweb.org/anthology/D17-1047.

Xilun Chen, Yu Sun, Ben Athiwaratkun, Claire Cardie, and Kilian Weinberger. Adversarial deep averaging networks for cross-lingual sentiment classification. *Transactions of the Association for Computational Linguistics*, 6:557–570, 2018. doi: 10.1162/tacl_a_00039. URL https://www.aclweb.org/anthology/Q18-1039.

Heeryon Cho, Songkuk Kim, Jongseo Lee, and Jong-Seok Lee. Data-driven integration of multiple sentiment dictionaries for lexicon-based sentiment classification of product reviews. *Know.-Based Syst.*, 71(1):61–71, November 2014a. ISSN 0950-7051. doi: 10.1016/j.knosys.2014.06.001. URL https://doi.org/10.1016/j.knosys.2014.06.001.

Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder–decoder approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 103–111, Doha, Qatar, October 2014b. Association for Computational Linguistics. doi: 10.3115/v1/W14-4012. URL https://www.aclweb.org/anthology/W14-4012.

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. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar, October 2014c. Association for Computational Linguistics. doi: 10.3115/v1/D14-1179. URL https://www.aclweb.org/anthology/D14-1179.

Yejin Choi and Claire Cardie. Hierarchical sequential learning for extracting opinions and their attributes. In *Proceedings of the ACL 2010 Conference Short Papers*, pages 269–274, Uppsala, Sweden, July 2010. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/P10-2050.

Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. Gated feedback recurrent neural networks. In *International conference on machine learning*, pages 2067–2075, 2015.

Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, page 160–167, New York, NY, USA, 2008. Association for Computing Machinery. ISBN 9781605582054. doi: 10.1145/1390156.1390177. URL https://doi.org/10.1145/1390156.1390177.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *J. Mach. Learn. Res.*, 12(null):2493–2537, November 2011. ISSN 1532-4435.

Wikimedia Commons. File:plutchik-wheel.svg — wikimedia commons, the free media repository, 2020. URL https://commons.wikimedia.org/w/index.php?title=File:Plutchik-wheel.svg&oldid=386724898. [Online; accessed 18-March-2020].

Alexis Conneau and Guillaume Lample. Cross-lingual language model pretraining. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 7059–7069. Curran Associates, Inc., 2019. URL http://papers.nips.cc/paper/8928-cross-lingual-language-model-pretraining.pdf.

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.

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

Michael Crawford, Taghi M Khoshgoftaar, Joseph D Prusa, Aaron N Richter, and Hamzah Al Najada. Survey of review spam detection using machine learning techniques. *Journal of Big Data*, 2(1):23, 2015.

Fermín L Cruz, José A Troyano, Beatriz Pontes, and F Javier Ortega. Building layered, multilingual sentiment lexicons at synset and lemma levels. *Expert Systems with Applications*, 41(13):5984–5994, 2014.

Kia Dashtipour, Soujanya Poria, Amir Hussain, Erik Cambria, Ahmad YA Hawalah, Alexander Gelbukh, and Qiang Zhou. Multilingual sentiment analysis: state of the art and independent comparison of techniques. *Cognitive computation*, 8(4):757–771, 2016.

Dmitry Davidov, Oren Tsur, and Ari Rappoport. Semi-supervised recognition of sarcasm in twitter and Amazon. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*, pages 107–116, Uppsala, Sweden, July 2010. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/W10-2914.

Bart Desmet and VéRonique Hoste. Emotion detection in suicide notes. *Expert Systems with Applications*, 40(16):6351–6358, 2013.

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. doi: 10.18653/v1/N19-1423. URL https://www.aclweb.org/anthology/N19-1423.

Xiaowen Ding, Bing Liu, and Philip S. Yu. A holistic lexicon-based approach to opinion mining. In *Proceedings of the 2008 International Conference on Web Search and Data Mining*, WSDM '08, page 231–240, New York, NY, USA, 2008. Association for Computing Machinery. ISBN 9781595939272. doi: 10.1145/1341531.1341561. URL https://doi.org/10.1145/1341531.1341561.

Xiaowen Ding, Bing Liu, and Lei Zhang. Entity discovery and assignment for opinion mining applications. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '09, page 1125–1134, New York, NY, USA, 2009. Association for Computing Machinery. ISBN 9781605584959. doi: 10.1145/1557019.1557141. URL https://doi.org/10.1145/1557019.1557141.

Hai Ha Do, P. W. C. Prasad, Angelika Maag, and Abeer Alsadoon. Deep learning for aspect-based sentiment analysis: A comparative review. *Expert Syst. Appl.*, 118:272–299, 2019.

Li Dong, Furu Wei, Chuanqi Tan, Duyu Tang, Ming Zhou, and Ke Xu. Adaptive recursive neural network for target-dependent twitter sentiment classification. In *The 52nd Annual Meeting of the Association for Computational Linguistics (ACL)*. ACL, 2014.

Xin Dong and Gerard De Melo. Cross-lingual propagation for deep sentiment analysis. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

Long Duong, Hiroshi Kanayama, Tengfei Ma, Steven Bird, and Trevor Cohn. Learning crosslingual word embeddings without bilingual corpora. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1285–1295, Austin, Texas, November 2016. Association for Computational Linguistics. doi: 10.18653/v1/D16-1136. URL https://www.aclweb.org/anthology/D16-1136.

Venkatesh Duppada, Royal Jain, and Sushant Hiray. SeerNet at SemEval-2018 task 1: Domain adaptation for affect in tweets. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 18–23, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/S18-1002. URL https://www.aclweb.org/anthology/S18-1002.

Sergey Edunov, Myle Ott, Michael Auli, and David Grangier. Understanding back-translation at scale. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 489–500, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1045. URL https://www.aclweb.org/anthology/D18-1045.

Paul Ekman. An argument for basic emotions. *Cognition & emotion*, 6(3-4): 169–200, 1992.

Jeffrey L Elman. Finding structure in time. *Cognitive science*, 14(2):179–211, 1990.

Manaal Faruqui and Chris Dyer. Improving vector space word representations using multilingual correlation. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 462–471, Gothenburg, Sweden, April 2014. Association for Computational Linguistics. doi: 10.3115/v1/E14-1049. URL https://www.aclweb.org/anthology/E14-1049.

Ronen Feldman. Techniques and applications for sentiment analysis. *Communications of the ACM*, 56(4):82–89, 2013.

Ronen Feldman, Benjamin Rosenfeld, Roy Bar-Haim, and Moshe Fresko. The stock sonar—sentiment analysis of stocks based on a hybrid approach. In *Twenty-third IAAI conference*, 2011.

John R Firth. A synopsis of linguistic theory, 1930-1955. *Studies in linguistic analysis*, 1957.

Nico Henri Frijda. The laws of emotion. *The American psychologist*, 43(5): 349–358, 1988.

Murthy Ganapathibhotla and Bing Liu. Mining opinions in comparative sentences. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 241–248, Manchester, UK, August 2008. Coling 2008 Organizing Committee. URL https://www.aclweb.org/anthology/C08-1031.

Aitor García-Pablos, Montse Cuadros, and German Rigau. W2vlda: Almost unsupervised system for aspect based sentiment analysis. *Expert Systems with Applications*, 91:127 – 137, 2018. ISSN 0957-4174. doi: https://doi.org/10.1016/j.eswa.2017.08.049. URL http://www.sciencedirect.com/science/article/pii/S0957417417305961.

Anastasia Giachanou and Fabio Crestani. Like it or not: A survey of twitter sentiment analysis methods. *ACM Computing Surveys (CSUR)*, 49(2):28, 2016.

Anne Laure Gilet, Daniel Grühn, Joseph Studer, and Gisela Labouvie-Vief. Valence, arousal, and imagery ratings for 835 french attributes by young, middle-aged, and older adults: The french emotional evaluation list (feel). *Revue Européenne de Psychologie Appliquée/European Review of Applied Psychology*, 62(3):173–181, 2012.

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

Pranav Goel, Devang Kulshreshtha, Prayas Jain, and Kaushal Kumar Shukla. Prayas at EmoInt 2017: An ensemble of deep neural architectures for emotion intensity prediction in tweets. In *Proceedings of the 8th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 58–65, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/W17-5207. URL https://www.aclweb.org/anthology/W17-5207.

Gene Howard Golub and Christian Reinsch. Singular value decomposition and least squares solutions. *Numer. Math.*, 14(5):403–420, April 1970. ISSN 0029-599X. doi: 10.1007/BF02163027. URL https://doi.org/10.1007/BF02163027.

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

Stephan Gouws and Anders Søgaard. Simple task-specific bilingual word embeddings. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1386–1390, Denver, Colorado, May–June 2015. Association for Computational Linguistics. doi: 10.3115/v1/N15-1157. URL https://www.aclweb.org/anthology/N15-1157.

Alex Graves and Jürgen Schmidhuber. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks*, 18(5-6):602–610, 2005.

Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*, pages 6645–6649. IEEE, 2013.

Marc Guasch, Pilar Ferré, and Isabel Fraga. Spanish norms for affective and lexico-semantic variables for 1,400 words. *Behavior research methods*, 48(4):1358–1369, 2016.

Jiang Guo, Wanxiang Che, David Yarowsky, Haifeng Wang, and Ting Liu. Cross-lingual dependency parsing based on distributed representations. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1234–1244, Beijing, China, July 2015. Association for Computational Linguistics. doi: 10.3115/v1/P15-1119. URL https://www.aclweb.org/anthology/P15-1119.

Samir Gupta, A.S.M. Ashique Mahmood, Karen Ross, Cathy Wu, and K. Vijay-Shanker. Identifying comparative structures in biomedical text. In *BioNLP 2017*, pages 206–215, Vancouver, Canada,, August 2017. Association for Computational Linguistics. doi: 10.18653/v1/W17-2326. URL https://www.aclweb.org/anthology/W17-2326.

Ivan Habernal, Tomáš Ptáček, and Josef Steinberger. Sentiment analysis in Czech social media using supervised machine learning. In *Proceedings of the 4th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 65–74, Atlanta, Georgia, June 2013. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/W13-1609.

Zellig S Harris. Distributional structure. *Word*, 10(2-3):146–162, 1954.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

Tomáš Hercig, Tomáš Brychcín, Lukáš Svoboda, and Michal Konkol. UWB at SemEval-2016 task 5: Aspect based sentiment analysis. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 342–349, San Diego, California, June 2016. Association for Computational Linguistics. doi: 10.18653/v1/S16-1055. URL https://www.aclweb.org/anthology/S16-1055.

Karl Moritz Hermann and Phil Blunsom. Multilingual distributed representations without word alignment, 2013.

John Hewitt. Finding syntax with structural probes, April 2019. URL https://nlp.stanford.edu//~johnhew//structural-probe.html?utm_source=quora&utm_medium=referral#the-structural-probe.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November 1997. ISSN 0899-7667. doi: 10.1162/neco.1997.9.8.1735. URL https://doi.org/10.1162/neco.1997.9.8.1735.

Jeremy Howard and Sebastian Ruder. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 328–339, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-1031. URL https://www.aclweb.org/anthology/P18-1031.

Minqing Hu and Bing Liu. Mining and summarizing customer reviews. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '04, page 168–177, New York, NY, USA, 2004. Association for Computing Machinery. ISBN 1581138881. doi: 10.1145/1014052.1014073. URL https://doi.org/10.1145/1014052.1014073.

Jiangping Huang, Chunli Xiang, Shuwei Yuan, Desen Yuan, and Xiaorui Huang. Character-aware convolutional recurrent networks with self-attention for emotion detection on twitter. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2019.

Kejun Huang, Matt Gardner, Evangelos Papalexakis, Christos Faloutsos, Nikos Sidiropoulos, Tom Mitchell, Partha P. Talukdar, and Xiao Fu. Translation invariant word embeddings. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1084–1088, Lisbon, Portugal, September 2015. Association for Computational Linguistics. doi: 10.18653/v1/D15-1127. URL https://www.aclweb.org/anthology/D15-1127.

Sarthak Jain and Shashank Batra. Cross lingual sentiment analysis using modified BRAE. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 159–168, Lisbon, Portugal, September 2015. Association for Computational Linguistics. doi: 10.18653/v1/D15-1016. URL https://www.aclweb.org/anthology/D15-1016.

Niklas Jakob and Iryna Gurevych. Extracting opinion targets in a single and cross-domain setting with conditional random fields. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1035–1045, Cambridge, MA, October 2010. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/D10-1101.

Wei Jin and Hung Hay Ho. A novel lexicalized hmm-based learning framework for web opinion mining. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, page

465–472, New York, NY, USA, 2009. Association for Computing Machinery. ISBN 9781605585161. doi: 10.1145/1553374.1553435. URL https://doi.org/10.1145/1553374.1553435.

Nitin Jindal and Bing Liu. Identifying comparative sentences in text documents. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '06, pages 244–251, New York, NY, USA, 2006a. ACM. ISBN 1-59593-369-7. doi: 10.1145/1148170.1148215. URL http://doi.acm.org/10.1145/1148170.1148215.

Nitin Jindal and Bing Liu. Mining comparative sentences and relations. In *Proceedings of the 21st National Conference on Artificial Intelligence - Volume 2*, AAAI'06, pages 1331–1336. AAAI Press, 2006b. ISBN 978-1-57735-281-5. URL http://dl.acm.org/citation.cfm?id=1597348.1597400.

Nitin Jindal and Bing Liu. Review spam detection. In *Proceedings of the 16th international conference on World Wide Web*, pages 1189–1190, 2007.

Nitin Jindal and Bing Liu. Opinion spam and analysis. In *Proceedings of the 2008 International Conference on Web Search and Data Mining*, WSDM '08, page 219–230, New York, NY, USA, 2008. Association for Computing Machinery. ISBN 9781595939272. doi: 10.1145/1341531.1341560. URL https://doi.org/10.1145/1341531.1341560.

Daniel Jurafsky and James H. Martin. *Speech and Language Processing (2nd Edition)*. Prentice-Hall, Inc., USA, 2009. ISBN 0131873210.

Christopher Kennedy. Comparatives, semantics of. *Concise Encyclopedia of Philosophy of Language and Linguistics*, pages 68–71, 2004.

Talaat Khalil and Samhaa R. El-Beltagy. NileTMRG at SemEval-2016 task 5: Deep convolutional neural networks for aspect category and sentiment extraction. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 271–276, San Diego, California, June 2016. Association for Computational Linguistics. doi: 10.18653/v1/S16-1043. URL https://www.aclweb.org/anthology/S16-1043.

Farhan Hassan Khan, Saba Bashir, and Usman Qamar. Tom: Twitter opinion mining framework using hybrid classification scheme. *Decision Support Systems*, 57:245 – 257, 2014. ISSN 0167-9236. doi: https://doi.org/10.1016/j.dss.2013.09.004. URL http://www.sciencedirect.com/science/article/pii/S0167923613002327.

Farhan Hassan Khan, Usman Qamar, and Saba Bashir. Swims: Semi-supervised subjective feature weighting and intelligent model selection for sentiment analysis. *Knowledge-Based Systems*, 100:97 – 111, 2016. ISSN 0950-7051. doi: https://doi.org/10.1016/j.knosys.2016.02.011. URL http://www.sciencedirect.com/science/article/pii/S0950705116000976.

Vinh Ngoc Khuc, Chaitanya Shivade, Rajiv Ramnath, and Jay Ramanathan. Towards building large-scale distributed systems for twitter sentiment analysis. In *Proceedings of the 27th Annual ACM Symposium on Applied Computing*, SAC '12, page 459–464, New York, NY, USA, 2012. Association for Computing Machinery. ISBN 9781450308571. doi: 10.1145/2245276.2245364. URL https://doi.org/10.1145/2245276.2245364.

Soo-Min Kim and Eduard Hovy. Determining the sentiment of opinions. In *Proceedings of the 20th International Conference on Computational Linguistics*, COLING '04, Stroudsburg, PA, USA, 2004. Association for Computational Linguistics. doi: 10.3115/1220355.1220555. URL https://doi.org/10.3115/1220355.1220555.

Soo-Min Kim and Eduard Hovy. Identifying and analyzing judgment opinions. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 200–207, New York City, USA, June 2006. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/N06-1026.

Yoon Kim. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar, October 2014. Association for Computational Linguistics. doi: 10.3115/v1/D14-1181. URL https://www.aclweb.org/anthology/D14-1181.

Svetlana Kiritchenko, Xiaodan Zhu, and Saif M. Mohammad. Sentiment analysis of short informal texts. *J. Artif. Int. Res.*, 50(1):723–762, May 2014. ISSN 1076-9757.

Roman Klinger, Orphée De Clercq, Saif Mohammad, and Alexandra Balahur. IEST: WASSA-2018 implicit emotions shared task. In *Proceedings of the 9th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 31–42, Brussels, Belgium, October 2018. Association for Computational Linguistics. doi: 10.18653/v1/W18-6206. URL https://www.aclweb.org/anthology/W18-6206.

Philipp Koehn. Europarl: A parallel corpus for statistical machine translation. In *MT summit*, volume 5, pages 79–86. Citeseer, 2005.

Olga Kolchyna, Tharsis T. P. Souza, Philip Treleaven, and Tomaso Aste. Twitter sentiment analysis: Lexicon method, machine learning method and their combination, 2015.

Maximilian Köper, Evgeny Kim, and Roman Klinger. IMS at EmoInt-2017: Emotion intensity prediction with affective norms, automatically extended resources and deep learning. In *Proceedings of the 8th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 50–57, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/W17-5206. URL https://www.aclweb.org/anthology/W17-5206.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, page 282–289, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc. ISBN 1558607781.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. ALBERT: A lite BERT for self-supervised learning of language representations. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=H1eA7AEtvS.

Stanislas Lauly, Alex Boulanger, and Hugo Larochelle. Learning multilingual word representations using a bag-of-words autoencoder. *CoRR*, abs/1401.1803, 2014. URL http://arxiv.org/abs/1401.1803.

Angeliki Lazaridou, Georgiana Dinu, and Marco Baroni. Hubness and pollution: Delving into cross-space mapping for zero-shot learning. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 270–280, Beijing, China, July 2015. Association for Computational Linguistics. doi: 10.3115/v1/P15-1027. URL https://www.aclweb.org/anthology/P15-1027.

Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. BioBERT: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36(4):1234–1240, 09 2019. ISSN 1367-4803. doi: 10.1093/bioinformatics/btz682. URL https://doi.org/10.1093/bioinformatics/btz682.

Fangtao Li, Minlie Huang, and Xiaoyan Zhu. Sentiment analysis with global topics and local dependency. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence*, AAAI'10, page 1371–1376. AAAI Press, 2010a.

Shasha Li, Chin-Yew Lin, Young-In Song, and Zhoujun Li. Comparable entity mining from comparative questions. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 650–658, Uppsala, Sweden, July 2010b. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/P10-1067.

Bing Liu. *Web Data Mining: Exploring Hyperlinks, Contents, and Usage Data (Data-Centric Systems and Applications)*. Springer-Verlag, Berlin, Heidelberg, 2006. ISBN 3540378812.

Bing Liu. Sentiment analysis and opinion mining. *Synthesis lectures on human language technologies*, 5(1):1–167, 2012.

Bing Liu et al. Sentiment analysis and subjectivity. *Handbook of natural language processing*, 2(2010):627–666, 2010.

Fei Liu, Trevor Cohn, and Timothy Baldwin. Recurrent entity networks with delayed memory update for targeted aspect-based sentiment analysis. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 278–283, New Orleans, Louisiana, June 2018a. Association for Computational Linguistics. doi: 10.18653/v1/N18-2045. URL https://www.aclweb.org/anthology/N18-2045.

Peter J. Liu, Mohammad Saleh, Etienne Pot, Ben Goodrich, Ryan Sepassi, Lukasz Kaiser, and Noam Shazeer. Generating wikipedia by summarizing long sequences. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018b. URL https://openreview.net/forum?id=Hyg0vbWC-.

Chong Long, Jie Zhang, and Xiaoyan Zhu. A review selection approach for accurate feature rating estimation. In *Coling 2010: Posters*, pages 766–774, Beijing, China, August 2010. Coling 2010 Organizing Committee. URL https://www.aclweb.org/anthology/C10-2088.

Bin Lu, Chenhao Tan, Claire Cardie, and Benjamin K. Tsou. Joint bilingual sentiment classification with unlabeled parallel corpora. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 320–330, Portland, Oregon, USA, June 2011. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/P11-1033.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis.

In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/P11-1015.

Christopher Manning, Prabhakar Raghavan, and Hinrich Schütze. Introduction to information retrieval. *Natural Language Engineering*, 16(1): 100–103, 2010.

Mika Mäntylä, Bram Adams, Giuseppe Destefanis, Daniel Graziotin, and Marco Ortu. Mining valence, arousal, and dominance: possibilities for detecting burnout and productivity? In *Proceedings of the 13th International Conference on Mining Software Repositories*, pages 247–258, 2016.

Mika V Mäntylä, Daniel Graziotin, and Miikka Kuutila. The evolution of sentiment analysis—a review of research topics, venues, and top cited papers. *Computer Science Review*, 27:16–32, 2018.

Justin Martineau and Timothy W. Finin. Delta tfidf: An improved feature space for sentiment analysis. In *ICWSM*, 2009.

Jonathan May, Ekaterina Shutova, Aurelie Herbelot, Xiaodan Zhu, Marianna Apidianaki, and Saif M. Mohammad, editors. *Proceedings of the 13th International Workshop on Semantic Evaluation*, Minneapolis, Minnesota, USA, June 2019. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/S19-2000.

Diana Maynard and Adam Funk. Automatic detection of political opinions in tweets. In *Extended Semantic Web Conference*, pages 88–99. Springer, 2011.

Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4): 115–133, 1943.

Walaa Medhat, Ahmed Hassan, and Hoda Korashy. Sentiment analysis algorithms and applications: A survey. *Ain Shams engineering journal*, 5 (4):1093–1113, 2014.

Qiaozhu Mei, Xu Ling, Matthew Wondra, Hang Su, and ChengXiang Zhai. Topic sentiment mixture: Modeling facets and opinions in weblogs. In *Proceedings of the 16th International Conference on World Wide Web*, WWW '07, page 171–180, New York, NY, USA, 2007. Association for Computing Machinery. ISBN 9781595936547. doi: 10.1145/1242572.1242596. URL https://doi.org/10.1145/1242572.1242596.

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

Tomas Mikolov, Quoc V. Le, and Ilya Sutskever. Exploiting similarities among languages for machine translation. *CoRR*, abs/1309.4168, 2013b. URL http://arxiv.org/abs/1309.4168.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, page 3111–3119, Red Hook, NY, USA, 2013c. Curran Associates Inc.

George A Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

George A Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine J Miller. Introduction to wordnet: An on-line lexical database. *International journal of lexicography*, 3(4):235–244, 1990.

Samaneh Moghaddam and Martin Ester. Opinion digger: An unsupervised opinion miner from unstructured product reviews. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*, CIKM '10, page 1825–1828, New York, NY, USA, 2010. Association for Computing Machinery. ISBN 9781450300995. doi: 10.1145/1871437.1871739. URL https://doi.org/10.1145/1871437.1871739.

Saif Mohammad. Obtaining reliable human ratings of valence, arousal, and dominance for 20,000 english words. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 174–184, 2018.

Saif Mohammad and Felipe Bravo-Marquez. WASSA-2017 shared task on emotion intensity. In *Proceedings of the 8th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 34–49, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/W17-5205. URL https://www.aclweb.org/anthology/W17-5205.

Saif Mohammad and Peter Turney. Emotions evoked by common words and phrases: Using mechanical turk to create an emotion lexicon. In *Proceedings of the NAACL HLT 2010 Workshop on Computational Approaches to Analysis and Generation of Emotion in Text*, pages 26–34, Los Angeles, CA, June 2010. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/W10-0204.

Saif Mohammad, Cody Dunne, and Bonnie Dorr. Generating high-coverage semantic orientation lexicons from overtly marked words and a thesaurus. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2-Volume 2*, pages 599–608. Association for Computational Linguistics, 2009.

Saif Mohammad, Felipe Bravo-Marquez, Mohammad Salameh, and Svetlana Kiritchenko. SemEval-2018 task 1: Affect in tweets. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 1–17, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/S18-1001. URL https://www.aclweb.org/anthology/S18-1001.

Saif M. Mohammad and Svetlana Kiritchenko. Using hashtags to capture fine emotion categories from tweets. *Computational Intelligence*, 31(2): 301–326, 2015. ISSN 1467-8640. doi: 10.1111/coin.12024. URL http://dx.doi.org/10.1111/coin.12024.

Saif M. Mohammad and Peter D. Turney. Crowdsourcing a word-emotion association lexicon. 29(3):436–465, 2013.

M Dolores Molina-González, Eugenio Martínez-Cámara, María-Teresa Martín-Valdivia, and José M Perea-Ortega. Semantic orientation for polarity classification in spanish reviews. *Expert Systems with Applications*, 40(18):7250–7257, 2013.

Catherine Monnier and Arielle Syssau. Affective norms for french words (fan). *Behavior research methods*, 46(4):1128–1137, 2014.

Agnes Moors, Jan De Houwer, Dirk Hermans, Sabine Wanmaker, Kevin Van Schie, Anne-Laura Van Harmelen, Maarten De Schryver, Jeffrey De Winne, and Marc Brysbaert. Norms of valence, arousal, dominance, and age of acquisition for 4,300 dutch words. *Behavior research methods*, 45(1):169–177, 2013.

Andrius Mudinas, Dell Zhang, and Mark Levene. Combining lexicon and learning based approaches for concept-level sentiment analysis. In *Proceedings of the First International Workshop on Issues of Sentiment Discovery and Opinion Mining*, WISDOM '12, New York, NY, USA, 2012. Association for Computing Machinery. ISBN 9781450315432. doi: 10.1145/2346676.2346681. URL https://doi.org/10.1145/2346676.2346681.

Arjun Mukherjee and Bing Liu. Aspect extraction through semi-supervised modeling. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 339–348, Jeju Island, Korea, July 2012. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/P12-1036.

Andrew Y. Ng and Michael I. Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, pages 841–848. MIT Press, 2002.

Vincent Ng, Sajib Dasgupta, and S. M. Niaz Arifin. Examining the role of linguistic knowledge sources in the automatic identification and classification of reviews. In *Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*, pages 611–618, Sydney, Australia, July 2006. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/P06-2079.

Finn Årup Nielsen. A new ANEW: evaluation of a word list for sentiment analysis in microblogs. In Matthew Rowe, Milan Stankovic, Aba-Sah Dadzie, and Mariann Hardey, editors, *Proceedings of the ESWC2011 Workshop on 'Making Sense of Microposts': Big things come in small packages, Heraklion, Crete, Greece, May 30, 2011*, volume 718 of *CEUR Workshop Proceedings*, pages 93–98. CEUR-WS.org, 2011. URL http://ceur-ws.org/Vol-718/paper_16.pdf.

Alvaro Ortigosa, José M. Martín, and Rosa M. Carro. Sentiment analysis in facebook and its application to e-learning. *Computers in Human Behavior*, 31:527 – 541, 2014. ISSN 0747-5632. doi: https://doi.org/10.1016/j.chb.2013.05.024. URL http://www.sciencedirect.com/science/article/pii/S0747563213001751.

Charles Egerton Osgood, George J Suci, and Percy H Tannenbaum. *The measurement of meaning*. Number 47. University of Illinois press, 1957.

Alexander Pak and Patrick Paroubek. Twitter as a corpus for sentiment analysis and opinion mining. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, Valletta, Malta, May 2010. European Language Resources Association (ELRA). URL http://www.lrec-conf.org/proceedings/lrec2010/pdf/385_Paper.pdf.

Girish K. Palshikar, Manoj Apte, Deepak Pandita, and Vikram Singh. Learning to identify subjective sentences. In *Proceedings of the 13th International Conference on Natural Language Processing*, pages 239–248, Varanasi, India, December 2016. NLP Association of India. URL https://www.aclweb.org/anthology/W16-6330.

S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2010.

Bo Pang and Lillian Lee. A sentimental education: Sentiment analysis using subjectivity. In *Proceedings of ACL*, pages 271–278, 2004.

Bo Pang and Lillian Lee. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 115–124, Ann Arbor, Michigan, June 2005. Association for Computational Linguistics. doi: 10.3115/1219840.1219855. URL https://www.aclweb.org/anthology/P05-1015.

Bo Pang and Lillian Lee. Opinion mining and sentiment analysis. *Found. Trends Inf. Retr.*, 2(1–2):1–135, January 2008. ISSN 1554-0669. doi: 10.1561/1500000011. URL https://doi.org/10.1561/1500000011.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. Thumbs up? sentiment classification using machine learning techniques. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP 2002)*, pages 79–86. Association for Computational Linguistics, July 2002. doi: 10.3115/1118693.1118704. URL https://www.aclweb.org/anthology/W02-1011.

W Gerrod Parrott. *Emotions in social psychology: Essential readings*. Psychology Press, 2001.

Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, October 2014. Association for Computational Linguistics. doi: 10.3115/v1/D14-1162. URL https://www.aclweb.org/anthology/D14-1162.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1202. URL https://www.aclweb.org/anthology/N18-1202.

Jakub Piskorski, Laska Laskova, Michał Marcińczuk, Lidia Pivovarova, Pavel Přibáň, Josef Steinberger, and Roman Yangarber. The second cross-lingual challenge on recognition, normalization, classification, and linking of named entities across Slavic languages. In *Proceedings of the 7th Workshop on Balto-Slavic Natural Language Processing*, pages 63–74, Florence, Italy, August 2019. Association for Computational Linguistics. doi: 10.18653/v1/W19-3709. URL https://www.aclweb.org/anthology/W19-3709.

Robert Plutchik. A general psychoevolutionary theory of emotion. In *Theories of emotion*, pages 3–33. Elsevier, 1980.

Marco Polignano, Pierpaolo Basile, Marco de Gemmis, and Giovanni Semeraro. A comparison of word-embeddings in emotion detection from text using bilstm, cnn and self-attention. In *Adjunct Publication of the 27th Conference on User Modeling, Adaptation and Personalization*, UMAP'19 Adjunct, page 63–68, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450367110. doi: 10.1145/3314183.3324983. URL https://doi.org/10.1145/3314183.3324983.

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. doi: 10.3115/v1/S14-2004. URL https://www.aclweb.org/anthology/S14-2004.

Maria Pontiki, Dimitris Galanis, Haris Papageorgiou, Ion Androutsopoulos, Suresh Manandhar, Mohammad AL-Smadi, Mahmoud Al-Ayyoub, Yanyan Zhao, Bing Qin, Orphée De Clercq, Véronique Hoste, Marianna Apidianaki, Xavier Tannier, Natalia Loukachevitch, Evgeniy Kotelnikov, Nuria Bel, Salud María Jiménez-Zafra, and Gülşen Eryiğit. SemEval-2016 task 5: Aspect based sentiment analysis. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 19–30, San Diego, California, June 2016. Association for Computational Linguistics. doi: 10.18653/v1/S16-1002. URL https://www.aclweb.org/anthology/S16-1002.

Ana-Maria Popescu and Oren Etzioni. Extracting product features and opinions from reviews. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 339–346, Vancouver, British Columbia, Canada, October 2005. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/H05-1043.

Soujanya Poria, Erik Cambria, and Alexander Gelbukh. Aspect extraction for opinion mining with a deep convolutional neural network. *Know.-Based Syst.*, 108(C):42–49, September 2016. ISSN 0950-7051. doi: 10.1016/j.knosys.2016.06.009. URL https://doi.org/10.1016/j.knosys.2016.06.009.

Ondřej Pražák, Pavel Přibáň, Stephen Taylor, and Jakub Sido. Uwb at semeval-2020 task 1: Lexical semantic change detection. In *Proceedings of the 14th International Workshop on Semantic Evaluation (SemEval-2020)*, Barcelona, Spain, Sep 2020. Association for Computational Linguistics.

Pavel Přibáň and Jiří Martínek. UWB at IEST 2018: Emotion prediction in tweets with bidirectional long short-term memory neural network. In *Proceedings of the 9th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 224–230, Brussels, Belgium, October 2018. Association for Computational Linguistics. doi: 10.18653/v1/W18-6232. URL https://www.aclweb.org/anthology/W18-6232.

Pavel Přibáň, Tomáš Hercig, and Ladislav Lenc. UWB at SemEval-2018 task 1: Emotion intensity detection in tweets. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 133–140, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/S18-1018. URL https://www.aclweb.org/anthology/S18-1018.

Pavel Přibáň, Tomáš Hercig, and Josef Steinberger. Machine learning approach to fact-checking in west Slavic languages. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2019)*, pages 973–979, Varna, Bulgaria, September 2019. INCOMA Ltd. doi: 10.26615/978-954-452-056-4_113. URL https://www.aclweb.org/anthology/R19-1113.

Tomáš Ptáček, Ivan Habernal, and Jun Hong. Sarcasm detection on Czech and English twitter. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 213–223, Dublin, Ireland, August 2014. Dublin City University and Association for Computational Linguistics. URL https://www.aclweb.org/anthology/C14-1022.

Lawrence R. Rabiner. *A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition*, page 267–296. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1990. ISBN 1558601244.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. *URL https://s3-us-west-2. amazonaws. com/openai-assets/researchcovers/languageunsupervised/language understanding paper. pdf*, 2018.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

Miloš Radovanović, Alexandros Nanopoulos, and Mirjana Ivanović. Hubs in space: Popular nearest neighbors in high-dimensional data. *J. Mach. Learn. Res.*, 11:2487–2531, December 2010. ISSN 1532-4435.

Delip Rao and Deepak Ravichandran. Semi-supervised polarity lexicon induction. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 675–682. Association for Computational Linguistics, 2009.

Shebuti Rayana and Leman Akoglu. Collective opinion spam detection: Bridging review networks and metadata. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '15, page 985–994, New York, NY, USA, 2015. Association for Computing Machinery. ISBN 9781450336642. doi: 10.1145/2783258.2783370. URL https://doi.org/10.1145/2783258.2783370.

Jaime Redondo, Isabel Fraga, Isabel Padrón, and Montserrat Comesaña. The spanish adaptation of anew (affective norms for english words). *Behavior research methods*, 39(3):600–605, 2007.

Yafeng Ren and Donghong Ji. Neural networks for deceptive opinion spam detection. *Inf. Sci.*, 385(C):213–224, April 2017. ISSN 0020-0255. doi: 10.1016/j.ins.2017.01.015. URL https://doi.org/10.1016/j.ins.2017.01.015.

Antonio Reyes, Paolo Rosso, and Davide Buscaldi. From humor recognition to irony detection: The figurative language of social media. *Data Knowl. Eng.*, 74:1–12, April 2012. ISSN 0169-023X. doi: 10.1016/j.datak.2012.02.005. URL https://doi.org/10.1016/j.datak.2012.02.005.

Monika Riegel, Małgorzata Wierzba, Marek Wypych, Łukasz Żurawski, Katarzyna Jednoróg, Anna Grabowska, and Artur Marchewka. Nencki affective word list (nawl): The cultural adaptation of the berlin affective word list–reloaded (bawl-r) for polish. *Behavior Research Methods*, 47(4): 1222–1236, 2015.

Ellen Riloff and Janyce Wiebe. Learning extraction patterns for subjective expressions. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*, pages 105–112, 2003. URL https://www.aclweb.org/anthology/W03-1014.

Ellen Riloff, Siddharth Patwardhan, and Janyce Wiebe. Feature subsumption for opinion analysis. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 440–448, Sydney, Australia, July 2006. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/W06-1652.

Kirk Roberts, Michael A. Roach, Joseph Johnson, Josh Guthrie, and Sanda M. Harabagiu. EmpaTweet: Annotating and detecting emotions on twitter. In *Proceedings of the Eighth International Conference on*

Language Resources and Evaluation (LREC'12), pages 3806–3813, Istanbul, Turkey, May 2012. European Language Resources Association (ELRA). URL http://www.lrec-conf.org/proceedings/lrec2012/pdf/201_Paper.pdf.

Pilar Rodriguez, Alvaro Ortigosa, and Rosa M Carro. Extracting emotions from texts in e-learning environments. In *2012 Sixth International Conference on Complex, Intelligent, and Software Intensive Systems*, pages 887–892. IEEE, 2012.

Sara Rosenthal, Noura Farra, and Preslav Nakov. SemEval-2017 task 4: Sentiment analysis in twitter. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 502–518, Vancouver, Canada, August 2017. Association for Computational Linguistics. doi: 10.18653/v1/S17-2088. URL https://www.aclweb.org/anthology/S17-2088.

Sebastian Ruder, Ivan Vulić, and Anders Søgaard. A survey of cross-lingual word embedding models. *J. Artif. Int. Res.*, 65(1):569–630, May 2019. ISSN 1076-9757. doi: 10.1613/jair.1.11640. URL https://doi.org/10.1613/jair.1.11640.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. *Learning Internal Representations by Error Propagation*, page 318–362. MIT Press, Cambridge, MA, USA, 1986. ISBN 026268053X.

James A Russell. A circumplex model of affect. *Journal of personality and social psychology*, 39(6):1161, 1980.

James A Russell. Core affect and the psychological construction of emotion. *Psychological review*, 110(1):145, 2003.

Marzieh Saeidi, Guillaume Bouchard, Maria Liakata, and Sebastian Riedel. SentiHood: Targeted aspect based sentiment analysis dataset for urban neighbourhoods. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1546–1556, Osaka, Japan, December 2016. The COLING 2016 Organizing Committee. URL https://www.aclweb.org/anthology/C16-1146.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. In *NeurIPS EMC Workshop*, 2019.

Christopher Scaffidi, Kevin Bierhoff, Eric Chang, Mikhael Felker, Herman Ng, and Chun Jin. Red opal: Product-feature scoring from reviews. In *Proceedings of the 8th ACM Conference on Electronic Commerce*, EC '07, page 182–191, New York, NY, USA, 2007. Association for Computing

Machinery. ISBN 9781595936530. doi: 10.1145/1250910.1250938. URL https://doi.org/10.1145/1250910.1250938.

Klaus R Scherer and Harald G Wallbott. Evidence for universality and cultural variation of differential emotion response patterning. *Journal of personality and social psychology*, 66(2):310, 1994.

David S Schmidtke, Tobias Schröder, Arthur M Jacobs, and Markus Conrad. Angst: Affective norms for german sentiment terms, derived from the affective norms for english words. *Behavior research methods*, 46(4): 1108–1118, 2014.

Hendrik Schuff, Jeremy Barnes, Julian Mohme, Sebastian Padó, and Roman Klinger. Annotation, modelling and analysis of fine-grained emotions on a stance and sentiment detection corpus. In *Proceedings of the 8th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 13–23, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/W17-5203. URL https://www.aclweb.org/anthology/W17-5203.

Mike Schuster and Kuldip K Paliwal. Bidirectional recurrent neural networks. *IEEE transactions on Signal Processing*, 45(11):2673–2681, 1997.

David A Shamma, Lyndon Kennedy, and Elizabeth F Churchill. Tweet the debates: understanding community annotation of uncollected sources. In *Proceedings of the first SIGMM workshop on Social media*, pages 3–10. ACM, 2009.

Bei Shi, Zihao Fu, Lidong Bing, and Wai Lam. Learning domain-sensitive and sentiment-aware word embeddings. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2494–2504, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-1232. URL https://www.aclweb.org/anthology/P18-1232.

Tianze Shi, Zhiyuan Liu, Yang Liu, and Maosong Sun. Learning cross-lingual word embeddings via matrix co-factorization. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 567–572, Beijing, China, July 2015. Association for Computational Linguistics. doi: 10.3115/v1/P15-2093. URL https://www.aclweb.org/anthology/P15-2093.

Kush Shrivastava, Shishir Kumar, and Deepak Kumar Jain. An effective approach for emotion detection in multimedia text data using sequence based convolutional neural network. *Multimedia Tools and Applications*, 78(20):29607–29639, 2019.

Lei Shu, Hu Xu, and Bing Liu. Lifelong learning CRF for supervised aspect extraction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 148–154, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-2023. URL https://www.aclweb.org/anthology/P17-2023.

Nikhil Kumar Singh, Deepak Singh Tomar, and Arun Kumar Sangaiah. Sentiment analysis: a review and comparative analysis over social media. *Journal of Ambient Intelligence and Humanized Computing*, pages 1–21, 2018.

Prerana Singhal and Pushpak Bhattacharyya. Borrow a little from your rich cousin: Using embeddings and polarities of English words for multilingual sentiment classification. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 3053–3062, Osaka, Japan, December 2016. The COLING 2016 Organizing Committee. URL https://www.aclweb.org/anthology/C16-1287.

Ana Paula Soares, Montserrat Comesaña, Ana P Pinheiro, Alberto Simões, and Carla Sofia Frade. The adaptation of the affective norms for english words (anew) for european portuguese. *Behavior research methods*, 44(1): 256–269, 2012.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA, October 2013. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/D13-1170.

Carina Söderholm, Emilia Häyry, Matti Laine, and Mira Karrasch. Valence and arousal ratings for 420 finnish nouns by age and gender. *PloS one*, 8 (8):e72859, 2013.

Michael Speriosu, Nikita Sudan, Sid Upadhyay, and Jason Baldridge. Twitter polarity classification with label propagation over lexical links and the follower graph. In *Proceedings of the First workshop on Unsupervised Learning in NLP*, pages 53–63. Association for Computational Linguistics, 2011.

Ralf Steinberger, Martin Atkinson, Teófilo Garcia, Erik Van der Goot, Jens Linge, Charles Macmillan, Hristo Tanev, Marco Verile, and Gerhard Wagner. Emm: Supporting the analyst by turning multilingual text into structured data. *Transparenz aus Verantwortung: Neue Herausforderungen für die digitale Datenanalyse*, 2017.

Philip J Stone, Dexter C Dunphy, and Marshall S Smith. The general inquirer: A computer approach to content analysis. 1966.

Carlo Strapparava and Rada Mihalcea. SemEval-2007 task 14: Affective text. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, pages 70–74, Prague, Czech Republic, June 2007. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/S07-1013.

Carlo Strapparava and Rada Mihalcea. Learning to identify emotions in text. In *Proceedings of the 2008 ACM symposium on Applied computing*, pages 1556–1560, 2008.

Chi Sun, Luyao Huang, and Xipeng Qiu. Utilizing BERT for aspect-based sentiment analysis via constructing auxiliary sentence. 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 380–385, Minneapolis, Minnesota, June 2019a. Association for Computational Linguistics. doi: 10.18653/v1/N19-1035. URL https://www.aclweb.org/anthology/N19-1035.

Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. How to fine-tune bert for text classification? In *China National Conference on Chinese Computational Linguistics*, pages 194–206. Springer, 2019b.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 3104–3112. Curran Associates, Inc., 2014. URL http://papers.nips.cc/paper/5346-sequence-to-sequence-learning-with-neural-networks.pdf.

Duyu Tang, Furu Wei, Nan Yang, Ming Zhou, Ting Liu, and Bing Qin. Learning sentiment-specific word embedding for twitter sentiment classification. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1555–1565. Association for Computational Linguistics, 2014. doi: 10.3115/v1/P14-1146. URL http://aclweb.org/anthology/P14-1146.

Mike Thelwall, Kevan Buckley, and Georgios Paltoglou. Sentiment strength detection for the social web. *JASIST*, 63(1):163–173, 2012. doi: 10.1002/asi.21662. URL https://doi.org/10.1002/asi.21662.

Ivan Titov and Ryan McDonald. Modeling online reviews with multi-grain topic models. In *Proceedings of the 17th International Conference on World Wide Web*, WWW '08, page 111–120, New York, NY, USA, 2008. Association for Computing Machinery. ISBN 9781605580852. doi: 10.1145/1367497.1367513. URL https://doi.org/10.1145/1367497.1367513.

Peter Turney. Thumbs up or thumbs down? semantic orientation applied to unsupervised classification of reviews. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 417–424, Philadelphia, Pennsylvania, USA, July 2002. Association for Computational Linguistics. doi: 10.3115/1073083.1073153. URL https://www.aclweb.org/anthology/P02-1053.

Frederik Vaassen. Measuring emotion. *Exploring the feasibility of automatically classifying emotional text*, 2014.

Alessandro Valitutti, Carlo Strapparava, and Oliviero Stock. Developing affective lexical resources. *PsychNology Journal*, 2(1):61–83, 2004.

Cynthia Van Hee, Els Lefever, and Véronique Hoste. SemEval-2018 task 3: Irony detection in English tweets. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 39–50, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/S18-1005. URL https://www.aclweb.org/anthology/S18-1005.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc., 2017. URL http://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf.

Julio Villena-Román. Tass 2013—workshop on sentiment analysis at sepln 2013: An overview. In *Proceedings of the TASS workshop at SEPLN*, pages 112–125, 2013.

Melissa LH Vo, Markus Conrad, Lars Kuchinke, Karolina Urton, Markus J Hofmann, and Arthur M Jacobs. The berlin affective word list reloaded (bawl-r). *Behavior research methods*, 41(2):534–538, 2009.

Ivan Vulic and Marie-Francine Moens. Bilingual distributed word representations from document-aligned comparable data. *J. Artif. Int. Res.*, 55 (1):953–994, January 2016. ISSN 1076-9757.

Yogarshi Vyas and Marine Carpuat. Sparse bilingual word representations for cross-lingual lexical entailment. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1187–1197, San Diego, California, June 2016. Association for Computational Linguistics. doi: 10.18653/v1/N16-1142. URL https://www.aclweb.org/anthology/N16-1142.

Xiaojun Wan. Using bilingual knowledge and ensemble techniques for unsupervised Chinese sentiment analysis. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 553–561, Honolulu, Hawaii, October 2008. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/D08-1058.

Xiaojun Wan. Co-training for cross-lingual sentiment classification. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 235–243, Suntec, Singapore, August 2009. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/P09-1027.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. 2019. In the Proceedings of ICLR.

Jin Wang, Liang-Chih Yu, K. Robert Lai, and Xuejie Zhang. Dimensional sentiment analysis using a regional CNN-LSTM model. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 225–230, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-2037. URL https://www.aclweb.org/anthology/P16-2037.

Amy Beth Warriner, Victor Kuperman, and Marc Brysbaert. Norms of valence, arousal, and dominance for 13,915 english lemmas. *Behavior research methods*, 45(4):1191–1207, 2013.

Paul J Werbos. Generalization of backpropagation with application to a recurrent gas market model. *Neural networks*, 1(4):339–356, 1988.

Janyce Wiebe. Learning subjective adjectives from corpora. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*, page 735–740. AAAI Press, 2000. ISBN 0262511126.

Janyce Wiebe and Ellen Riloff. Creating subjective and objective sentence classifiers from unannotated texts. In *Proceedings of the 6th International Conference on Computational Linguistics and Intelligent Text Processing*, CICLing'05, page 486–497, Berlin, Heidelberg, 2005. Springer-Verlag. ISBN 3540245235. doi: 10.1007/978-3-540-30586-6_53. URL https://doi.org/10.1007/978-3-540-30586-6_53.

Janyce Wiebe and Theresa Wilson. Learning to disambiguate potentially subjective expressions. In *COLING-02: The 6th Conference on Natural Language Learning 2002 (CoNLL-2002)*, 2002. URL https://www.aclweb.org/anthology/W02-2034.

Janyce Wiebe, Rebecca F Bruce, and Thomas P O'Hara. Development and use of a gold-standard data set for subjectivity classifications. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics*, pages 246–253, 1999.

Janyce Wiebe, Theresa Wilson, and Claire Cardie. Annotating expressions of opinions and emotions in language. *Language resources and evaluation*, 39(2-3):165–210, 2005.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. Recognizing contextual polarity in phrase-level sentiment analysis. In *HLT/EMNLP 2005, Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference, 6-8 October 2005, Vancouver, British Columbia, Canada*, pages 347–354. The Association for Computational Linguistics, 2005. URL https://www.aclweb.org/anthology/H05-1044/.

Fu Xianghua, Liu Guo, Guo Yanyan, and Wang Zhiqiang. Multi-aspect sentiment analysis for chinese online social reviews based on topic modeling and hownet lexicon. *Know.-Based Syst.*, 37:186–195, January 2013. ISSN 0950-7051. doi: 10.1016/j.knosys.2012.08.003. URL https://doi.org/10.1016/j.knosys.2012.08.003.

Min Xiao and Yuhong Guo. Distributed word representation learning for cross-lingual dependency parsing. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning*, pages 119–129, Ann Arbor, Michigan, June 2014. Association for Computational Linguistics. doi: 10.3115/v1/W14-1613. URL https://www.aclweb.org/anthology/W14-1613.

Hu Xu, Bing Liu, Lei Shu, and Philip S. Yu. Double embeddings and CNN-based sequence labeling for aspect extraction. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume*

2: Short Papers), pages 592–598, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-2094. URL https://www.aclweb.org/anthology/P18-2094.

Seon Yang and Youngjoong Ko. Extracting comparative entities and predicates from texts using comparative type classification. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1636–1644, Portland, Oregon, USA, June 2011. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/P11-1164.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 5753–5763. Curran Associates, Inc., 2019.

Liang-Chih Yu, Lung-Hao Lee, Shuai Hao, Jin Wang, Yunchao He, Jun Hu, K Robert Lai, and Xuejie Zhang. Building chinese affective resources in valence-arousal dimensions. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 540–545, 2016.

Liang-Chih Yu, Jin Wang, K. Robert Lai, and Xuejie Zhang. Refining word embeddings for sentiment analysis. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 534–539, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/D17-1056. URL https://www.aclweb.org/anthology/D17-1056.

Dongwen Zhang, Hua Xu, Zengcai Su, and Yunfeng Xu. Chinese comments sentiment classification based on word2vec and svmperf. *Expert Systems with Applications*, 42(4):1857 – 1863, 2015a. ISSN 0957-4174. doi: https://doi.org/10.1016/j.eswa.2014.09.011. URL http://www.sciencedirect.com/science/article/pii/S0957417414005508.

Lei Zhang, Riddhiman Ghosh, Mohamed Dekhil, Meichun Hsu, and Bing Liu. Combining lexicon-based and learning-based methods for twitter sentiment analysis. 01 2011.

Lei Zhang, Shuai Wang, and Bing Liu. Deep learning for sentiment analysis: A survey. *WIREs Data Mining and Knowledge Discovery*, 8(4):e1253, 2018. doi: 10.1002/widm.1253. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/widm.1253.

Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, page 649–657, Cambridge, MA, USA, 2015b. MIT Press.

Han Zhao, Zhengdong Lu, and Pascal Poupart. Self-adaptive hierarchical sentence model. In *Proceedings of the 24th International Conference on Artificial Intelligence*, IJCAI'15, page 4069–4076. AAAI Press, 2015. ISBN 9781577357384.

Xin Zhao, Jing Jiang, Hongfei Yan, and Xiaoming Li. Jointly modeling aspects and opinions with a MaxEnt-LDA hybrid. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 56–65, Cambridge, MA, October 2010. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/D10-1006.

HuiWei Zhou, Long Chen, Fulin Shi, and Degen Huang. Learning bilingual sentiment word embeddings for cross-language sentiment classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 430–440, Beijing, China, July 2015. Association for Computational Linguistics. doi: 10.3115/v1/P15-1042. URL https://www.aclweb.org/anthology/P15-1042.

Xinjie Zhou, Xiaojun Wan, and Jianguo Xiao. Attention-based LSTM network for cross-lingual sentiment classification. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 247–256, Austin, Texas, November 2016. Association for Computational Linguistics. doi: 10.18653/v1/D16-1024. URL https://www.aclweb.org/anthology/D16-1024.

Jingbo Zhu, Huizhen Wang, Benjamin K. Tsou, and Muhua Zhu. Multi-aspect opinion polling from textual reviews. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management*, CIKM '09, page 1799–1802, New York, NY, USA, 2009. Association for Computing Machinery. ISBN 9781605585123. doi: 10.1145/1645953.1646233. URL https://doi.org/10.1145/1645953.1646233.

Will Y. Zou, Richard Socher, Daniel Cer, and Christopher D. Manning. Bilingual word embeddings for phrase-based machine translation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1393–1398, Seattle, Washington, USA, October 2013. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/D13-1141.