Bachelor's Thesis

Czech
Technical
University
in Prague

F3

Faculty of Electrical Engineering
Department of Cybernetics

Advanced Sentiment Analysis

Stanislav Lamoš

Supervisor: Ing. Jan Pichl
May 2023

BACHELOR‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name:	**Lamoš Stanislav**	Personal ID number:	**499062**
Faculty / Institute:	**Faculty of Electrical Engineering**		
Department / Institute:	**Department of Cybernetics**		
Study program:	**Open Informatics**		
Specialisation:	**Artificial Intelligence and Computer Science**		

II. Bachelor’s thesis details

Bachelor’s thesis title in English:

Advanced Sentiment Analysis

Bachelor’s thesis title in Czech:

Pokročilé metody rozpoznávání sentimentu

Guidelines:

The thesis assignment is to analyze and implement sentiment analysis methods focused on various aspects.
Steps:
1. Review current methods used for sentiment analysis and named entity recognition, with a focus on supervised methods based on embedding clustering and Language Model (LM) methods.
2. Review the current datasets for sentiment analysis, aspect-based sentiment analysis and named entity recognition.
3. Select the suitable model architecture based on the reviewed method.
4. Modify one of the selected methods for aspect sentiment analysis and dependency sentiment analysis.
5. Test the proposed methods on selected datasets.
6. Discuss the results and compare them with SOTA in the field.

The thesis assignment covers various aspects of Sentiment Analysis and Machine Learning techniques, it will require a solid understanding of the field and the ability to implement, analyze and compare the results.

Bibliography / sources:

[1] DO, Hai Ha, et al. Deep learning for aspect-based sentiment analysis: a comparative review. Expert systems with applications, 2019, 118: 272-299.
[2] NADEAU, David; SEKINE, Satoshi. A survey of named entity recognition and classification. Lingvisticae Investigationes, 2007, 30.1: 3-26.
[3] MEDHAT, Walaa; HASSAN, Ahmed; KORASHY, Hoda. Sentiment analysis algorithms and applications: A survey. Ain Shams engineering journal, 2014, 5.4: 1093-1113.

Name and workplace of bachelor’s thesis supervisor:

Ing. Jan Pichl Department of Cybernetics FEE

Name and workplace of second bachelor’s thesis supervisor or consultant:

Date of bachelor’s thesis assignment: **02.02.2023** Deadline for bachelor thesis submission: **26.05.2023**

Assignment valid until: **22.09.2024**

Ing. Jan Pichl	prof. Ing. Tomáš Svoboda, Ph.D.	prof. Mgr. Petr Páta, Ph.D.
Supervisor’s signature	Head of department’s signature	Dean’s signature

III. Assignment receipt

The student acknowledges that the bachelor’s thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the bachelor’s thesis, the author must state the names of consultants and include a list of references.

.	
Date of assignment receipt	Student’s signature

Acknowledgements

I want to thank Ing. Jan Pichl and Ing. Jan Šedivý for giving me valuable advice. I also want to thank my family and friends for supporting me throughout my university studies.

Declaration

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.
Prague, May 24, 2023

Abstract

Sentiment analysis determines the sentiment polarity towards a given document, sentence or aspect. The output of such classification gives us the information about emotional tone of given text. In this thesis, we perform several experiments and we report results of sentence-level and document-level sentiment analysis, aspect category detection, and aspect category sentiment analysis. We also discuss the theoretical background of such methods and review state-of-the-art models and datasets. Even though our models achieve worse results compared to state-of-the-art, the common advantage of our approaches is low time and space complexity.

Keywords: natural language processing, sentiment analysis, aspect category sentiment analysis

Supervisor: Ing. Jan Pichl

Abstrakt

Analýza sentimentu určuje polaritu sentimentu vzhledem k danému dokumentu, větě či aspektu. Výstup této klasifikace nám dává informaci o emoční zabarvenosti daného textu. V této práci provádíme množství experimentů a reportujeme výsledky analýzy sentimentu na úrovni vět a dokumetů, detekce kategorie aspektů a analýzy sentimentu kategorie aspektů. Také popisujeme teoretickou stránku těchto metod a děláme průzkum současných modelů a datasetů. I když naše modely vykazují horší výsledky než současná řešení, mají nízkou časovou a paměťovou náročnost.

Klíčová slova: zpracování přirozeného jazyka, analýza sentimentu, analýza sentimentu kategorie aspektů

Překlad názvu: Pokročilé metody rozpoznávání sentimentu

Contents

Figures

Tables

Chapter 1

Introduction

In this thesis, we focus on the task of Sentiment analysis. We perform sentiment analysis on three levels: document-level sentiment analysis, sentence-level sentiment analysis and aspect-based sentiment analysis (aspect category sentiment analysis in particular). Also, we propose methods for such classification tasks. Our motivation to focus on Sentiment analysis stems from a wide range of applications of this task. Using models trained for sentiment analysis, we can determine the emotional tone of the given text. Information about the polarity of certain text can be used in conversational systems to generate more accurate responses from chatbots. In addition, sentiment analysis can be used to classify reviews on social media so brands can monitor how customers feel about them.

1.1 Goal of the thesis

The goal of the thesis is to analyze and implement methods for sentence-level, document-level, and aspect-based sentiment analysis. In both practical and theoretical parts of the thesis, we have followed these steps:

- Theoretical background of NLP
- Review of current methods used for sentiment analysis and named entity recognition
- Review the current datasets for sentiment analysis and named entity recognition
- Review state-of-the-art methods for sentiment analysis and named entity recognition
- Implementation of proposed methods for sentiment analysis which are time and memory-efficient, training on selected datasets
- Discussion of results and conclusion of the thesis

1.2 Structure of the thesis

This thesis is divided into six chapters:

1. ***Introduction***: Describes motivation and goals of this thesis.
2. ***Theoretical background***: Introduces basic terms, such as: Machine Learning or Natural Language Processing. Describes theory of algorithms used to approach Named Entity Recognition and Sentiment Analysis tasks. Explains various concepts of how to numerically represent words.
3. ***Problem description and related work***: Defines the problem we focus on in this thesis, both in theoretical and practical parts. Describes related work and available datasets.
4. ***Proposed methods***: Describes methods we implemented to solve various tasks of sentiment analysis.
5. ***Experiments***: Describes practical experiments with reference to datasets, algorithmic approaches and discusses achieved results.
6. ***Conclusion***: Reviews achieved results and used methods, concludes the whole thesis.

Chapter 2

Theoretical background

In this chapter, we describe the theoretical background of methods used for sentiment analysis and named entity recognition. Also, we provide a general overview of Machine Learning (ML) and Natural Language Processing (NLP).

2.1 Machine Learning

Machine Learning is the field of computer science that focuses on computational methods based on learning from experience. The experience means provided data in this context. Various statistical methods are trained on a corpus of data and subsequently used for the classification or clustering of new and unseen data samples. Ultimately, we try to train the model which gives the most accurate predictions and generalizes well over sample space.

The term Machine Learning was first used in 1959 by IBM employee Arthur Samuel as he proposed an algorithm to solve the game of checkers[1]. Since then, many techniques have been developed (such as: Decision trees, Support Vector Machines, or Neural Networks) which improved state-of-the-art results. Nowadays, the area of Machine Learning is rapidly growing and is widely used in various applications[2], such as:

- Image recognition used for face detection or traffic monitoring in self-driving cars
- Speech recognition to convert voice input to text and analyze it in order to produce the most relevant answer in chatbots
- Spam filtering in email inboxes using Naive Bayes or Multilayer Perceptron classifiers
- Product recommendations based on user interest and their purchase history

Every machine learning algorithm uses provided dataset to train the classification model. Each record in the dataset describes attributes of the object from the area of our interest. The span of these attributes is called feature space. Let $D = \{x_1...x_n\}$ be given dataset composed of n samples. Every sample is characterized by k features. Therefore, every instance is formalized as $x_i = \{x_{i1}...x_{ik}\}$. Also, each sample can be assigned with its outcome called label. Then, ith sample is written as tuple of (x_i, y_i). We split the data into two sets - training and test. We build our model using training set. The process is called training or learning. Afterwards, we evaluate the quality of the model on test set. The point is to train the model which best approximates the ground truth[3].

Whether the given data are labelled or not, there are four main types of machine learning:

1. **Supervised Learning** involves training a model using labelled data and making predictions based on that learning.
2. **Unsupervised Learning** is about training a model using unlabeled data, which means that the data has not been classified or labelled. Later, it is used in applications such as clustering, association or dimensionality reduction.
3. **Semi-Supervised Learning** trains a model on the combination of a small amount of labelled data with a large amount of unlabeled data.
4. **Reinforcement Learning** trains a model to make decisions based on feedback from its environment. The model learns to take actions that maximize a reward function.

2.2 Natural Language Processing

One subfield of computer science that heavily relies on machine learning is Natural Language Processing (NLP). NLP focuses on the interaction between computers and human language. It involves the development of algorithms and models that enable computers to understand, interpret, and generate human language. NLP involves a wide range of tasks, including language translation, text classification, named entity recognition, part-of-speech tagging, and sentiment analysis. Previously, research in this area focused on training individual models on specific tasks. In recent years, this paradigm has shifted away in favour of large language models. These deep learning models commonly use transformer architecture and are trained on a large text corpus. On the upside, these general-purpose models perform well on a wide range of NLP tasks. Unfortunately, they require a lot of memory and computational power and the whole training process costs millions of dollars.

All in all, NLP is a very promising industry with a lot of future challenges and opportunities. One of many reasons to be optimistic about the outlook of this field is the user interest and extensive media coverage of ChatGPT released by OpenAI in November 2022. To put it into perspective, Figure 2.1 shows the estimated market size of this field.

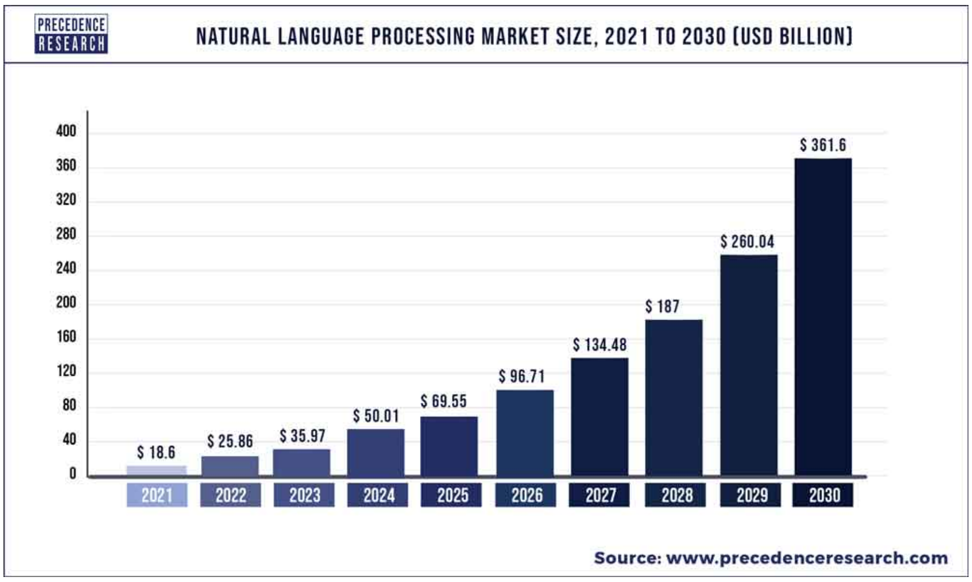

Figure 2.1: Estimated size of NLP market[4].

2.3 Word representations

In this section, we describe how to numerically represent words. Such representations treat words either as discrete symbols regardless of meaning, or as embedding vectors with encoded context using distributional semantics. Subsequently, numeric word representations are used as input for machine learning algorithms.

2.3.1 One-hot Vectors

The simplest way to represent words for ML models is to regard them as discrete symbols using one-hot vectors. The representation uses a word index from the provided vocabulary. It means that ith word from the vocabulary has one on ith index of vector and the rest of the dimensions are filled with zeros. The example is shown in 2.1 where vocabulary is $V = [cat, dog, lion, horse, monkey]$.

$$\begin{aligned} dog &= [0\,1\,0\,0\,0] \\ cat &= [1\,0\,0\,0\,0] \end{aligned} \tag{2.1}$$

As a vocabulary, we can use large lexical database named WordNet, published in [5], where we can find synonyms or hypernyms. WordNet is also available in Python NLTK package[1]. Unfortunately, vocabulary resources, such as WordNet, have a lot of problems. It is impossible to keep those databases up to date and they lack accurate computation of word similarity.

In general, one-hot vectors cannot compute the similarity of words. The result of the dot product of two different one-hot vectors is always orthogonal. Working with one-hot vectors is also computationally expensive because vector dimension equals the vocabulary size. Since one-hot vectors do not capture word meaning or context, it is a very inaccurate representation.

2.3.2 Word embeddings

A more complex method of representing words is to use word embeddings. Based on semantic theory, the meaning of the word is determined by words that frequently appear in the same context. In other words, *"You shall know a word by the company it keeps"*[6]. Word embeddings take the context of the words into account and thus accurately describe relationships among vectors with similar, or completely different meanings. As a result, we can compute the similarity between two embeddings using dot product or perform subtraction and addition operations. Also, embeddings have fewer dimensions than one-hot vectors, mostly between 100D and 1000D. The distance among embeddings in vector space corresponds to their closeness in terms of meaning. There have been multiple algorithms to generate word embeddings. Table 2.1 shows some examples.

Embedding framework	*Dimension*	*Associated paper*
fasttext	300	[7]
word2vec	300	[8, 9]
GloVe	25,50,100,200,300	[10]
USE	512	[11]
ELMo	1024	[12]

Table 2.1: Examples of embedding frameworks.

word2vec

One of the embedding frameworks is word2vec, published in [8] and [9]. Firstly, the algorithm constructs vocabulary from a large corpus of text. Then, it iterates over each word presented in the text corpus and treats each word as center word and context (or outside) word. The center word is the ith word at ith position in the text during ith iteration. The context words are surrounding words that appear within the fixed-size window. The size of the

[1]`https://www.nltk.org/howto/wordnet.html`

context window is an integer constant. Researchers behind word2vec came up with two models of training: Continuous Bag of Words (CBOW) and Skip-gram (SG).

CBOW predicts the center word given a bag of context words. Whereas, SG predicts the context words given the center word. The prediction is position independent. During the training process performed in multiple epochs, we keep adjusting the probabilities in word vectors in order to maximize them. We start with randomly initialized word vectors. The difference between CBOW and SG is shown in Figure 2.2.

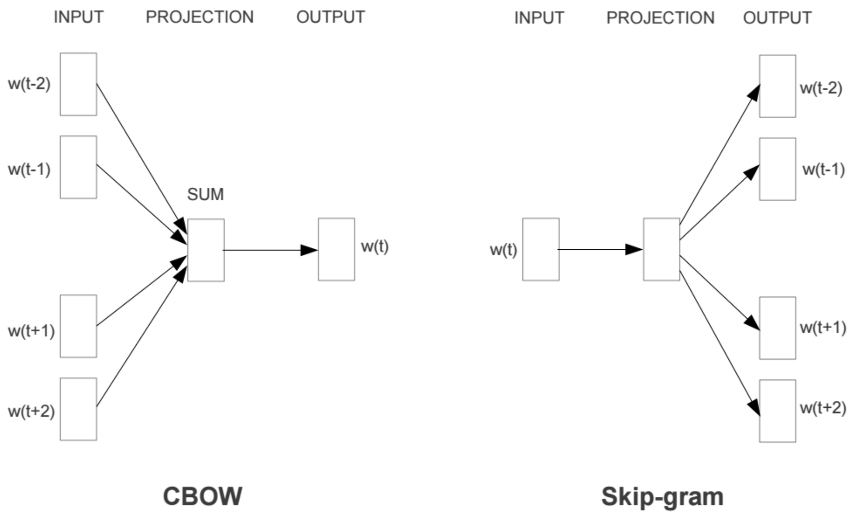

Figure 2.2: Difference between training models of word2vec[8].

Using SG, we predict outside words based on the center word. The objective of SG is to maximize log probability described in Equation 2.2. T is the size of training text corpus, w_t is center word and w_{t+j} is context word within fixed-size window c.

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c,\, j\neq 0}\log p(w_{t+j}|w_t) \tag{2.2}$$

Conditional probability using softmax function is defined in Equation 2.3 where c is the centre word and o is the outside (context) word. For each word, we have center vector v and context vector u. V is the dictionary size.

$$p(o|c) = \frac{\exp({u_o}^T v_c)}{\sum_{w\in V}\exp({u_w}^T v_c)} \tag{2.3}$$

At the end of training, we end up with two vectors for each word - context and center. We average them to obtain the final one.

Since computing softmax is a very expensive method, word2vec introduced two efficiency techniques for training: Hierarchical softmax and Negative sampling.

Hierarchical softmax. Hierarchical softmax uses binary Huffman tree (introduced in paper [13]). Each node can be reached from the root node using binary classification based on the sigmoid function. With W words, we only evaluate $\log W$ nodes. Therefore, the cost of computing power is lower.

Negative sampling. Negative sampling trains individual sigmoid functions for each word. It selects several negative samples and decreases their gradients.

fasttext

The idea of enriching embeddings with subword information was introduced in three similar publications: [14], [15] and [7]. Paper [7] came up with fasttext algorithm that incorporates n-gram features.

Fasttext is an extension of the word2vec algorithm. Word2vec treats each word as a general sequence of chars and represents it with a distinct vector. It does not take the morphology or semantic structure of words into account. As a result, word2vec performs poorly in morphologically rich languages where certain words appear rarely in the training corpora. However, fasttext represents each word by character n-grams[7].

For example, if we take the word *"under"* and $n = 3$, its n-gram representation would be: *<un, und, nde, der, er>*. Each n-gram representation starts and ends with special symbols *"<>"* to distinguish prefixes and suffixes. The final word vector is obtained as the sum of word embedding and embeddings of character n-grams. Formally, it is defined in Equation 2.4. Function s maps its scores of (word, context) to $\mathbb{R}$. G_w is the set of word's n-grams and v_c is the vector of context. z_g defines the vector representation of character n-grams[7].

$$s(w, c) = \sum_{g \in G_w} z_g^T v_c \tag{2.4}$$

Paper [16] used fasttext for classification tasks. It achieved state-of-the-art performance and higher accuracy on sentiment-analysis datasets than methods based on deep learning. The objective of the classification model is to minimize negative log-likelihood. Formally, it is defined in Equation 2.5.

$$-\frac{1}{N} \sum_{n=1}^{N} y_n \log\left(f(BAx_n)\right) \tag{2.5}$$

N is the set of documents, y_n is the label of nth document and x_n is the vector representation of nth document. The vector representation of each document is the average of its word embeddings. A and B are weight matrices and f is softmax function. The architecture of the model is similar to the CBOW method but fasttext classifier replaces the middle word with the label.

Universal Sentence Encoder (USE)

Paper [11] introduced two new models for sentence-level embeddings. They take an English string as input and output a 512-dimensional vector representation of given text. They were implemented in TensorFlow[17]. Training data of both encoders include supervised as well as unsupervised samples. Unsupervised data come from online sources, such as: Wikipedia, discussion forums or news pages. Annotated data were retrieved from Stanford Natural Language Inference (SNLI) corpus[18]. Paper [11] proposed two variants of embedding models. One is based on Transformer architecture[19] and the other uses deep averaging network (DAN)[20]. Using transformer-based, or DAN model is a trade-off between accuracy and efficiency.

Transformer-based USE creates embeddings by the encoder block of the Transformer model. The encoding layer takes input text and converts it into a representation with attention information. 512-dimensional sentence vector is obtained after computing the element-wise sum of each word. USE based on Transformer architecture achieves better accuracy but with longer computation time and higher memory requirements[11].

DAN-based USE works with deep averaging network. Feedforward neural network, fed with averaged word embeddings, generates 512-dimensional sentence embeddings. This variant of USE is more memory and time efficient but gives worse results thus resulting in lower accuracy[11].

2.4 Traditional machine learning models

The term "traditional machine learning models" refers to a collection of models characterized by their simple structure. These models can be used for addressing classification, regression, or clustering problems. The usage of the term "traditional machine learning" typically serves to contrast it with "deep learning", which denotes deep neural network models. Deep learning models exhibit a more complex architecture and are often employed for tasks requiring higher levels of abstraction and representation learning. The group of traditional machine learning models includes various algorithms, such as:

- Support vector machines (SVM)
- k-nearest neighbours (KNN)

- Logistic regression
- K-means
- Naive Bayes
- Conditional random fields (CRF)
- Hidden Markov model (HMM)

2.4.1 Logistic regression

Logistic regression is a statistical model whose objective is to train a classifier. Logistic regression learns on training samples by minimizing loss function, such as cross-entropy. Minimization of such loss function is done using stochastic gradient descent (SGD) that constantly adjusts parameters of the model, weights and biases, during the training process. After we have learned optimal weights and biases, we can make predictions on test set[21]. The mathematical formula is shown in Equation 2.6.

$$y = Xw + b \tag{2.6}$$

$X \in \mathbb{R}^{(n,c)}$ is the matrix of n test data with c features, $w \in \mathbb{R}^{(c,1)}$ is the column vector of weights and $b \in \mathbb{R}^{(n,1)}$ is the column vector of biases. The vector $y \in \mathbb{R}^{(n,1)}$, test data multiplied by weights and summed with biases, holds predicted labels for each vector from n input vectors.

In the case of binary classification, we compute the probability z of classifying sample x_i into class 1. The formula is shown in Equation 2.7.

$$z = \sigma(w \cdot x_i + b) \tag{2.7}$$

The probability of classifying x_i as class 0 can be obtained by $1 - z$. To make the final decision, we have to set a decision threshold and then determine the final label.

If we want to approach multiclass classification, we can use multinomial logistic regression which computes probabilities by softmax function. In general, logistic regression has various applications in classification tasks, for example: sentiment classification or part-of-speech tagging[21].

2.4.2 Hidden Markov model (HMM)

Hidden Markov model (HMM) is a statistical model characterized by two stochastic processes. The first one represents the Markov model with a finite set of states. In the case of HMM, states are hidden and thus unobservable. Such Markov model is described by a probability matrix of transitioning from one state to another called the transition probability matrix. Also, the initial probability matrix determines the probability of starting in a certain state. The second defining stochastic process is the observation process. Since states are hidden in HMM, they can be learned by observing observations that were generated by the most probable sequence of hidden states. Hidden states generate observations with probability distribution called emission probability. The objective of HMM is to find the most suitable sequence of hidden states that caused given observations. As shown in Table 2.2, HMMs are formalized by quintuple (S, A, π, B, O)[22].

$S = s_1 ... s_N$	set of N states
$A = a_{11} ... a_{N1} ... a_{NN}$	transition probability matrix
$\pi = \pi_1 ... \pi_N$	initial probability distribution
$O = o_1 ... o_T$	T observations
$B = b_i(o_t)$	emission probabilities

Table 2.2: Formalization of HMMs[21].

HMMs assume that the probability of appearing in the next hidden state depends exclusively on the current hidden state we are in. It is called Markov assumption[22]. The formula is shown in Equation 2.8.

$$P(s_i | s_1 ... s_{i-1}) = P(s_i | s_{i-1}) \tag{2.8}$$

Another assumption instantiated by HMM is the independence assumption. It says that the probability of current observation is solely determined by the last producing hidden state and not by any other states or observations[21]. The formal definition is shown in Equation 2.9.

$$P(o_i | s_1 ... s_i, ..., s_T, o_1, ..., o_i, ..., o_T) = P(o_i | s_i) \tag{2.9}$$

The transition probability is probability distribution specifying the probability of transitioning from one hidden state to another. Emission probability refers to the likelihood of observing certain outputs generated by a particular hidden state. Both transition and emission probabilities are parameters which are learned during the training process. To optimally adjust these parameters, we can use algorithms such as: Baum–Welch, forward-backward algorithm, or Expectation Maximization (EM)[21].

HMMs model the relationships among hidden states and output observations. The process of discovering a sequence of hidden states given observations is called decoding. To decode the sequence of hidden states, we use the Viterbi algorithm. Viterbi algorithm uses the principle of dynamic programming. In four steps (initialization, recursion, termination and backtracking), Viterbi finds the most probable sequence of hidden states given the observations[22].

Overall, HMMs can be useful architecture for various NLP tasks, for example: Named entity recognition (NER). In NER task, hidden states represent entity labels and observations are word tokens. Our goal is to find the most probable sequence of entity tags for a given text. Unfortunately, HMMs cannot capture long-range dependencies. The fact that they capture only a little context makes HMMs less accurate than more advanced machine learning methods.

2.4.3 Conditional random field (CRF)

Conditional random field (CRF) is a discriminative model for labelling sequential data. In this subsection, we focus on linear-chain CRF. It is the variant of CRF that has fewer computational requirements and is mainly used in NLP tasks.

Compared to HMMs, CRFs do not assume that the current state is solely dependent on the previous state. In general, we consider the concept of CRFs as more universal than HMMs. CRF determines the most probable sequence of labels Y with respect to the entire sequence of input words X. The objective of CRF is to classify all input words with the most optimal tags[21]. An example of CRF is shown in Figure 2.3.

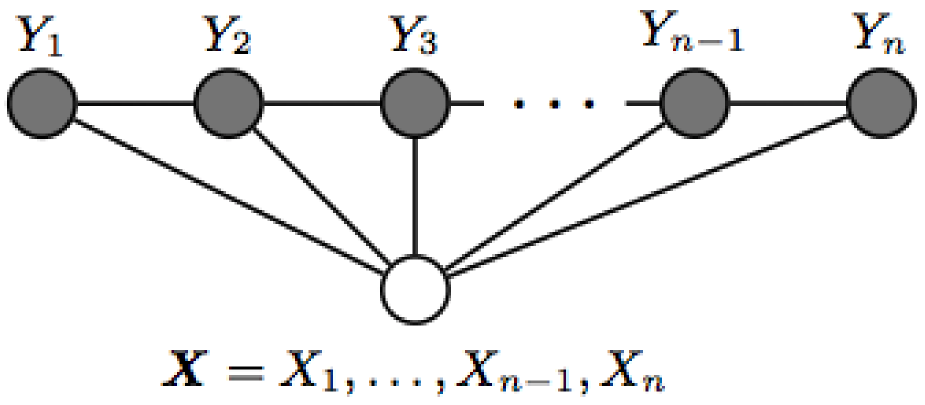

Figure 2.3: Example of CRFs[23].

Given the input words $X = x_1 ... x_N$, output tags $Y = y_1 ... y_N$ and possible tag sequences $\mathcal{Y}$, CRFs maximize the posterior probability $P = (Y|X)$ during training to determine the predictions[21]. The mathematical expression is shown in Equation 2.10.

$$\hat{Y} = \underset{Y \in \mathcal{Y}}{\arg\max}\, P(Y|X) \tag{2.10}$$

To model the relationships between input words and output labels, CRFs use feature functions. Feature functions extract relevant information from text. We can use various feature functions[24], such as:

- word length
- word suffix or prefix
- word embeddings

Posterior probability takes advantage of feature functions which map the sequence of X and Y to feature vector. The complete equation of finding the best tag sequence is in Equation 2.11.

$$\hat{Y} = \arg\max_{Y \in \mathcal{Y}} P(Y|X) = \arg\max_{Y \in \mathcal{Y}} \sum_{i=1}^{N} \sum_{k=1}^{K} w_k f_k(y_{i-1}, y_i, X, i) \tag{2.11}$$

As shown in Equation 2.11, the optimal sequence of labels is found by maximizing the sum over input words and the sum of weighted features. Each feature function f_k is multiplied by weight w_k. Feature functions, in the linear chain variant of CRF, take current tag y_i, previous tag y_{i-1}, input string X and timestamp index i on the input. As in the case of HMMs, we can use the forward-backward algorithm to adjust weights and the Viterbi algorithm to find the optimal sequence of labels[21].

Compared to HMMs, CRFs can capture long-range dependencies. Therefore, they can model relationships between words that appear far from each other in sentences. For example, we can use CRFs for part-of-speech tagging (POS) or named entity recognition (NER).

2.5 Neural Networks

Neural networks (NNs) are computational models based on the structure and function of the human brain. Neurons are basic units of the brain and nervous system. They are specialized cells that transmit information throughout the body in the form of electrical signals. Neurons are connected by synapses that transmit nerve impulses from one neuron to another. When a neuron receives a signal, it generates an electrical impulse that fires into other neurons. With a network composed of billions of neurons, the human brain is able to process information or control behaviour.

Neural networks model replaces neurons with nodes with activation functions and biases which are connected by edges with weights. These layers of interconnected nodes are used for various regression and classification tasks, such as: image recognition or text classification. An example of NN is shown in Figure 2.4.

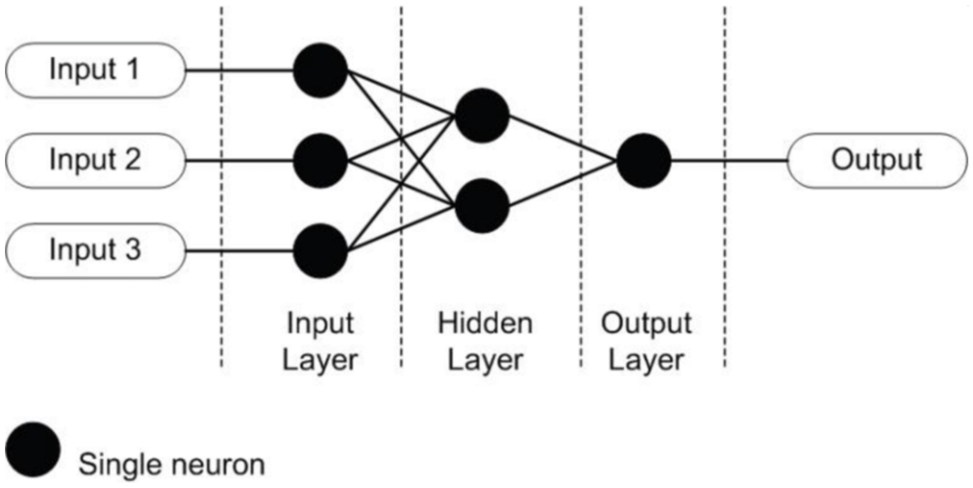

Figure 2.4: Example of NN architecture[25].

There are different types of neural networks with different architecture and preferred usage:

- Feedforward Neural Networks
- Convolutional Neural Networks (CNNs)
- Recurrent Neural Networks (RNNs)
- Long Short-Term Memory (LSTM) Networks
- Transformers

2.5.1 Feedforward Neural Networks

This is the simplest form of neural network. Information flows in only one direction, from input to output. Feedforward neural network is composed of the input layer, one or more hidden layers and the output layer. This architecture was introduced in paper [26]. Layers composed of neurons are fully connected by edges without cycles. It means that neurons take outputs from all neurons in the previous layer as inputs. Compared to previous methods, such architecture works with linearly nonseparable data. The objective of NNs is to minimize training loss. Loss is modelled using loss functions, such as: negative log-likelihood, mean squared error or binary cross-entropy. The output layer is the last layer of NNs and outputs probability distribution. According to paper [27], feedforward NNs with as few as one hidden layer can approximate any continuous functions.

The value of the loss function indicates how well our network fits the training data. Lower value indicates better performance. Each neuron has its activation function and bias. Edges connecting neurons have weights. On training examples, we compute weighted sums of outputs from the previous layer. Formally, weighted sum is defined in Equation 2.12 where w is a weight vector and x is an input vector.

$$z = w \cdot x \tag{2.12}$$

Weighted sum z is then summed with bias b and fed into the activation function associated with each neuron. The output of activation function σ in mth neuron is described in Equation 2.13.

$$a^m = \sigma(z^m + b^m) \tag{2.13}$$

After we complete forward propagation through all layers, we get the output probability distribution. We can use various activation functions in our models, such as:

- tanh
$$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$
- sigmoid
$$f(x) = \frac{1}{1 + e^{-x}}$$
- ReLU
$$f(x) = max(0, x)$$
- softplus
$$f(x) = \ln(1 + e^x)$$

The loss function models the distance between the output of NN and ground truth. To find parameters of such minimized loss function, we use a gradient descent algorithm. Gradient descent minimizes loss function by taking partial derivatives with respect to individual weights and biases. To determine optimal weights and biases, we use the backpropagation algorithm. The vector that contains partial derivatives of weights and biases is called a gradient. Gradient descent finds local minima of loss function by taking steps in size of learning rate in the negative direction of gradient [21]. Formally, it is defined in Equation 2.14. θ is parameter vector, f is differentiable function and η is learning rate.

$$\theta_{k+1} = \theta_k - \eta \nabla f(\theta_k) \tag{2.14}$$

Since performing gradient descent on all training samples and averaging their weights and biases is very slow, we use Stochastic gradient descent (SGD). In each training epoch, it selects a certain number of samples from randomly shuffled training data and finds parameters of loss function on those mini-batches.

In general, setting the learning rate has been recognized as tricky. SGD is very sensitive to the value of the learning rate. High learning rate results in divergence and NN with low learning rate converges slowly. We can use various optimizers to improve convergence to the optimal solution. Such optimizers are for example: Adam[28] or AdaGrad[29].

2.5.2 Recurrent Neural Networks

Recurrent neural network (RNN) is a type of NN which is an extension of simple feedforward neural network. RNN introduces the concept of time series data into the training process. While inputs and outputs in feedforward neural networks are independent of each other, RNN has memory through hidden layers to influence current input and output as it remembers calculations from prior outputs. RNNs share weights and biases across each layer thus these parameters are constantly adjusted during training. At each time step, prior outputs and current inputs are used to determine the current output. As in feedforward neural networks, weights and biases are optimized by SGD. Conversely, RNNs use backpropagation through time algorithm that takes time steps into account. An example of RNN architecture is shown in Figure 2.5.

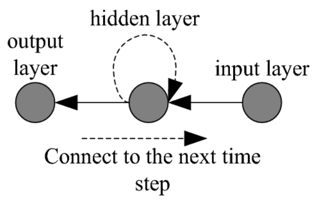

Figure 2.5: Example of RNN architecture[30].

RNNs are widely used for tasks in the field of natural language processing or speech recognition. However, RNNs suffer from vanishing and exploding gradient problems. Also, they tend to run into difficulties with long sequences of data. To partially fix these issues, more complex variants of RNNs have been proposed. In particular, Long Short-Term Memory (LSTM) and Gated recurrent unit (GRU) which use gated mechanisms. In most cases, LSTM and GRU perform similarly. But papers [31] and [32] showed examples where LSTM gives better results.

2.5.3 Transformers

Current state-of-the-art models in NLP are Transformers. Introduced in paper [19], their architecture is based on the Attention mechanism. The transformer-based models consist of two parts: encoder and decoder. An example of Transformer's internal structure is shown in Figure 2.6.

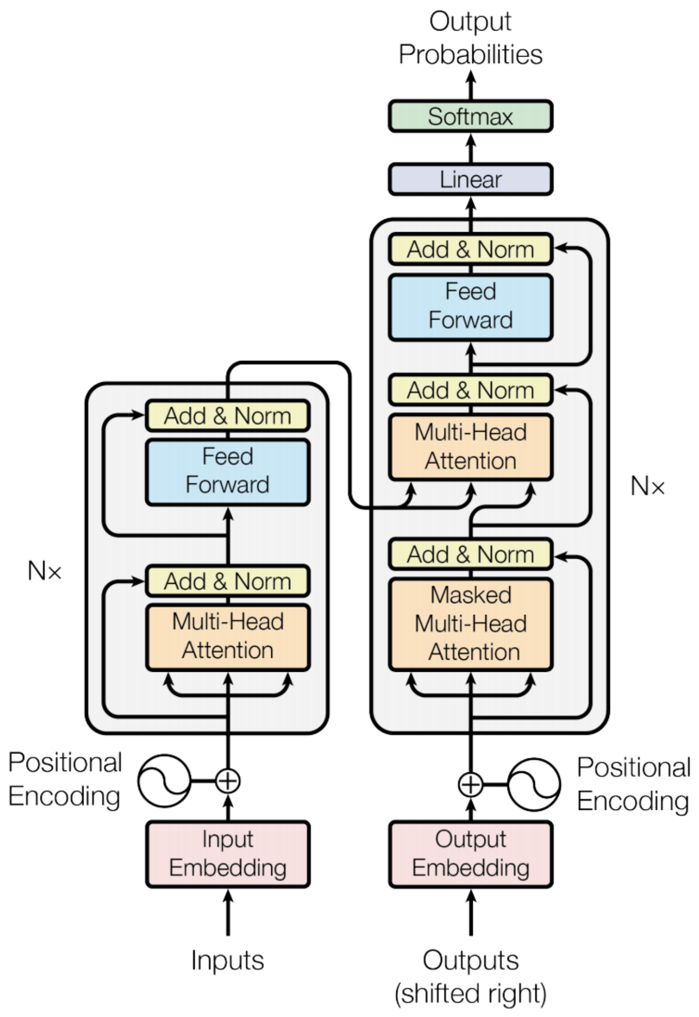

Figure 2.6: Architecture of transformer-based model[19].

Firstly, the input sequence is converted into positional input embeddings and fed into the encoder block. The encoding layer is composed of multi-head attention followed by a fully-connected layer. Multi-head attention uses self-attention to associate words from input with each other. The output from multi-head attention is then put into a feedforward network with ReLU as an activation function for further computation. In general, the encoder block encodes the input into a representation with attention information. Then, the decoder uses this information for the purpose of focusing on the appropriate words.

Similarly as the encoding layer, the decoding layer has two multi-head attention layers and a fully-connected feedforward network plus linear and softmax layers. The objective of the decoder block is to generate text sequences. The output of a fully-connected network goes to the linear classifier and softmax function that generates output probabilities.

Some of the widely used large language models were developed using transformer-based architecture. To be specific, some of them are: BERT[33], GPT-3[34] and RoBERTa[35].

Chapter 3

Problem description and related work

In this chapter, we describe the NLP tasks of Named entity recognition (NER) and Sentiment analysis (SA). We also introduce related work and available datasets for such tasks.

3.1 Sentiment Analysis

Sentiment analysis is a natural language processing technique used to determine the sentiment or emotional tone expressed in a piece of text, such as: a movie review, social media post or news article. Sentiment analysis can be applied to a wide range of applications[36], such as:

- Social media monitoring to find out what customers are saying about a particular brand and respond to negative comments
- Analyze and categorize customer reviews
- Analyze customer sentiment towards products, services, or brands to gain insights into consumer behaviour and preferences
- Monitor trending issues of customer support in order to assess its effectiveness

Sentiment Analysis (SA) is considered a classification task which determines attitude towards the given body of text. It can be divided into three levels so we can perform such classification of polarity on document, sentence or aspect level. Document-level SA takes a block of text, composed of multiple sentences, and identifies whether it is positive, or negative. Whilst, sentence-level SA decide whether the individual sentence is expressing positive, or negative feelings. Even though sentence-level SA provides a more detailed examination of emotional tone in a given text, there is no fundamental difference between these two classification levels[37].

The most thorough analysis of sentiment provides classification performed on aspect level. Aspect-level or aspect-based sentiment analysis (ABSA) focuses on sentiment analysis with respect to individual aspects of each sentence. Aspect can be a term explicitly mentioned in given sentence, or an implicit category predefined in the used dataset. Each sentence can contain multiple aspects with different polarities. Therefore, ABSA is a more challenging task compared to document-level and sentence-level SA.

3.1.1 Sentence-level and Document-level sentiment analysis

Sentence-level sentiment classification is the NLP task that analyzes sentiment in individual sentences. Algorithms, developed for this task, determine the overall polarity of opinions expressed in sentence samples. Polarity of expressed sentiment can be positive, or negative. Alternatively, it is either the three-way classification task which labels whether the emotional tone of a sentence is positive, negative, or neutral, or multiclass classification in general. The output of sentence-level sentiment analysis can be used to gain insights into how people feel about a particular topic or product[38].

Document-level sentiment analysis is also NLP task that analyzes sentiment. In this case, sentiment analysis is performed on document level. Document-level sentiment analysis takes blocks of text, composed of multiple sentences, and determines sentiment polarity labels. It could be binary classification with positive and negative tags, three-way classification with positive, negative and neutral classes, or multiclass classification in general.

Related work

Paper [39] proposed three models based on LSTM classifiers. These models differ in network architecture and parameters.

In [40], researchers introduced a model that implements four deep learning techniques. Firstly, sentences from the movie review dataset were encoded using word2vec. The architecture of the model was composed of Convolutional layer, Maxpooling layer, LSTM layer and Dense layer.

Paper [41] fine-tuned transformer-based model named RoBERTa. The experiment was performed on movie review dataset.

The classification model, published in [42], proposed an architecture where three state-of-the-art classifiers were trained: Naive Bayes, Support vector machines (SVM) and Maximum entropy. A trained classifier was then applied to test data from the Sentiment140 dataset to predict positive and negative labels. It also incorporated unigram and bigram features to improve the performance of the model.

Available datasets

We can find a large number of datasets for the tasks of sentence-level and document-level sentiment analysis. The dataset, published in [43] and created by Stanford University, is composed of 1.6 million tweets about products or brands. Tweets are labelled either as positive, or negative. The dataset from [44], obtained also by Stanford University, contains fifty thousand highly polar movie reviews. Reviews have either positive, or negative polarity. Paper [45] introduced the Stanford Sentiment Treebank dataset that contains movie reviews scrapped from the Rotten Tomatoes database. Models trained on this dataset try to predict five labels: negative, somewhat negative, neutral, positive and somewhat positive. Paper [46] constructed dataset for the task of Twitter sentiment analysis. Samples were manually annotated with positive, negative, or neutral labels. A thorough analysis of datasets, used in the experimental part of this thesis, is in sections 5.1.1 and 5.1.2.

3.1.2 Aspect-based sentiment analysis

Aspect-based sentiment analysis is the text classification task that determines the sentiment polarity with respect to aspects from the given sentence. This level of sentiment analysis gives the most detailed analysis of opinions as it tries to detect multiple aspects with different polarities. According to paper [47], the sentiment analysis is composed of two main elements: target g and sentiment s. Those two components generate tuple (g, s), where g is the aspect about which an opinion has been declared and s is the emotional tone of such opinion - positive, negative, or neutral.

Aspect is expressed either as an aspect term, or aspect category. Overall, the research of ABSA focuses on the detection, extraction or classification of four main attributes - aspect category, aspect term, opinion term and sentiment polarity. Aspect category represents a unique aspect which labels entities from text. Aspect categories come from the predefined set that corresponds to specific dataset. For example, FOOD and PRICE categories appear in data from restaurant domain. An aspect term is an entity explicitly mentioned in the text. Such entity is the target of opinion. In the sentence *"Iphone is expensive"*, the aspect term is *Iphone*. Opinion term is judgement expressing sentiment towards the target. The emotional tone of the opinion term determines the polarity of the target. In the sentence *"Iphone is expensive"*, the word *expensive* is the opinion term. Sentiment polarity is label that describes the attitude of opinion term towards the target. Mostly, sentiment polarity is one of positive, negative, or neutral[48]. Another example of ABSA classification is shown in Figure 3.1.

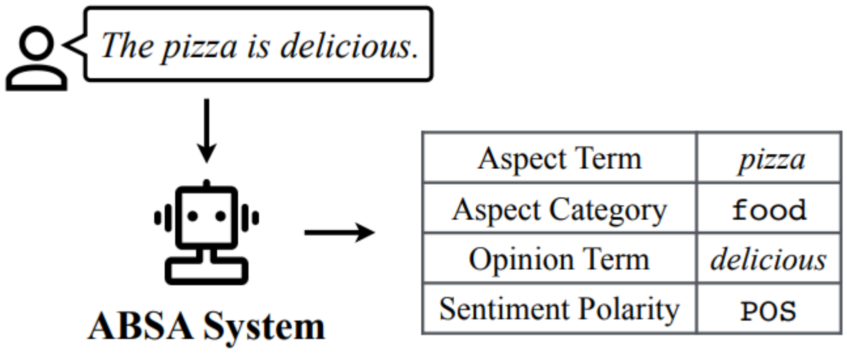

Figure 3.1: Example of ABSA classification[48].

ABSA is a complex task composed of multiple subtasks which are associated with the four main attributes defined above. There are three underlying subtasks: Opinion target extraction (OTE), Aspect category detection (ACD) and Sentiment polarity (SP). Opinion target extraction is concerned with labelling multiple aspect terms in sentences. Aspect category detection deals with identifying aspect categories that are mentioned in given text. Sentiment polarity task determines the positive, negative and neutral tags with respect to aspect terms or aspect categories.

In this thesis, we focus on aspect category and sentiment polarity pair extraction. For each sample from test set, we predict one or more pairs *(aspect category, sentiment polarity)*. Such task is called aspect category sentiment analysis (ACSA)[49]. Formally, let $s = [w_1, ..., w_n]$ be the sentence s composed of n words. Also, given a predefined set of m categories $C = \{c_1, ..., c_m\}$ and sentiment polarities $P = \{positive, negative, neutral\}$. The goal of ACSA is to predict $\{..., (y_c^i, y_p^i), ...\}$ for each sentence, where y_c^i is the ith aspect category and y_p^i is the ith sentiment polarity towards predicted aspect category[50].

Related work

Paper [51] proposed a new End-to-End Convolutional neural network (CNN) that jointly performed ACD and SP on GermEval 2017 dataset. Apart from classifying positive, negative and neutral polarity towards aspect category as done in previous papers, they added extra dimension *"N/A"* denoting whether the aspect category appears in the sentence, or not. As a result, they performed ACD and SP simultaneously with significant performance gains.

Model from [52] performed ACSA on the Chinese dataset. They used two CNNs to obtain a representation of each sample. The final embedding was input into multiple Multilayer perceptron (MLP) classifiers for joint learning.

Paper [50] presented a Hierarchical Graph Convolutional Network (Hier-GCN) as an approach to tackle ACSA. Lower-level GCN was used to detect aspect categories and higher-level GCN predicted corresponding sentiment polarities.

Approach from [53] transformed ACSA as a classification task into a language modelling task. They used language models to predict aspect category and sentiment polarity.

In paper [54], they proposed a model composed of five layers for joint classification of aspect categories and sentiment polarities. The layers are: the input layer, the shared BiLSTM layer, the special BiLSTM layer, the multiple perspective attention layer, and the multilabel classifier layer.

Paper [55] introduced a model based on AS-Capsules where hidden vectors are encoded by recurrent neural networks (RNN). The number of AS-Capsules equals to the number of aspect categories and each capsule outputs the probability of aspect category and sentiment polarity distribution.

Available datasets

The biggest disadvantage of this task is the lack of large datasets. In most of the existing papers, the SemEval datasets, released in papers [56], [57] and [58], are benchmarks. The data come from restaurant and laptop reviews. Paper [59] took the data from restaurant reviews for the period 2014-2016 and merged them. As a result, *"Restaurant-Large"* dataset was constructed. Also, paper [60] released dataset for ACSA which contains at least two aspect categories with different polarities per each sentence. A thorough analysis of datasets, used in the experimental part of this thesis, is in section 5.1.3.

3.2 Named Entity Recognition

Named entity recognition (NER) is a natural language processing (NLP) task that involves identifying and classifying named entities in text. Named entities are terms specifically mentioned in the text by name, such as: people, organizations, locations, or products. Firstly, NER models identify words that correspond to named entities. It means that they label whether the given word is named entity, or not. Then, NER systems classify named entities into predefined categories. For example, the NER system might identify the words *Tom Cruise* and *New York* in text and classify them as PERSON and LOCATION, respectively. We can use NER in a wide range of applications, for example: information extraction, knowledge base construction, and text summarization. It is also an important component of many natural language processing systems as named entities give key contextual information about the given text.

To predict labels of named entities in given dataset, we can use multiple algorithms. Most common sequence labelling algorithms divide into the following categories[61]:

- Rule-based methods using vocabularies or dictionaries.
- Feature-based supervised learning approaches such as: HMM or CRF.
- Deep learning techniques using machine learning models, for example: RNN or LSTM.

These three major categories are the most commonly used to tackle NER-related tasks. Particularly, Feature-based and deep learning approaches represent state-of-the-art technologies widely employed for extracting named entities from text with high accuracy.

The technique to label entities in given dataset is called IOB. IOB tagging, also known as Inside-Outside-Beginning tagging, is scheme for annotating text data with named entity labels. Using IOB tagging, each word in the text is assigned label that indicates whether it is the beginning of the named entity (B), inside of the named entity (I), or outside of the named entity (O). For example, consider the following text: *"Tom Cruise was born in New York"*. Words *Tom* and *Cruise* would be labelled as B-PER (beginning of PERSON entity) and I-PER (inside of PERSON entity), respectively. Likewise, word *New* would be labelled as B-LOC (beginning of a LOCATION entity). Lastly, the word *York* would be labelled as I-LOC (inside a LOCATION entity). The remaining words in the text would be labelled as O (outside of the named entity). Another example is shown in Figure 3.2.

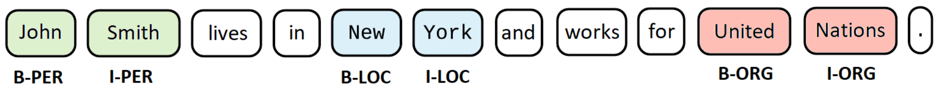

Figure 3.2: Example of IOB tagging[62].

3.2.1 Related work

Paper [63] proposed new BioALBERT model. This transformer-based model was created by fine-tuning ALBERT[64] on large text corpora from biomedical domain. It was tested on domain-specific datasets to extract named entities of diseases, chemicals and etc.

Model from [65] made use of neural network architecture composed of CNN, bidirectional LSTM and CRF. Character-level representations, computed by CNN and GloVe word embeddings, were concatenated and fed into bidirectional LSTM. The final label was obtained by CRF.

Classifier proposed in paper [66] used model from paper [67] and replaced widely used cross entropy loss with dice loss. Such solution tackles the problem of imbalanced datasets and achieves state-of-the-art results.

Model from [68] applied GRU and CRF on entity detection and classification. They performed two experiments: multi-task training and cross-lingual training. Multi-task training involves joint training of models for POS and NER tasks. The cross-lingual method involves joint training of model for one task (POS, or NER) in multiple languages.

In paper [69], BERT model was used to perform NER on Czech datasets.

3.2.2 Available datasets

Paper [70] created the NER dataset in two languages: English and German. Four types of named entities are recognized in the text. Data were taken from Reuters news articles. Paper [71] proposed dataset composed of broadcast news and web data. Eighteen entity types were defined in such dataset. Dataset from paper [72] include named entities annotated on Czech text. There are forty six entity types. Paper [73] introduced dataset composed of annotated data with four entity types. Data come from Wikipedia articles. Research published in paper [74] introduced dataset for NER task from the medical domain. It classifies entities into five predefined categories of proteins.

Chapter 4

Proposed methods

In this chapter, we describe methods we have used to obtain results in the experimental part of the thesis. Methods have been used for aspect category detection, aspect category sentiment analysis, document-level sentiment analysis and sentence-level sentiment analysis.

4.1 Fasttext supervised classification

According to paper [16], fasttext can be used for various text classification tasks. Reported results from paper [16] have outperformed methods based on deep learning techniques but with significantly lower computation costs and memory requirements. The efficiency and accuracy of the fasttext classifier are the main reasons why we have used fasttext in our experiments.

We were given sentences from our selected datasets. After performing text formatting and cleaning operations (data preprocessing is described in 5.3), we fed training samples into fasttext classifier. The sentence is tokenized using an internal fasttext tokenizer and represented as a set of N ngram features. Formally, training corpus is composed of K sentences $S = \{s_1...s_K\}$ and ith sentence consists of ngram features as $s_i = \{x_1...x_N\}$. Averaged representations of words are fed into linear classifier. The probability distribution over predefined labels is computed by softmax function. Figure 4.1 depicts the internal structure of fasttext classifier.

Regarding sentence-level sentiment analysis, each training sentence was labelled as positive, or negative. In case of document-level sentiment analysis, each training sample, composed of multiple sentences, has also either positive, or negative label. We did binary classification on those samples. Input data were in desired fasttext format. Example is shown in 4.1.

$$\begin{aligned} &\_\_label\_\_positive \quad i\ like\ chocolate \\ &\_\_label\_\_negative \quad i\ hate\ bananas \end{aligned} \tag{4.1}$$

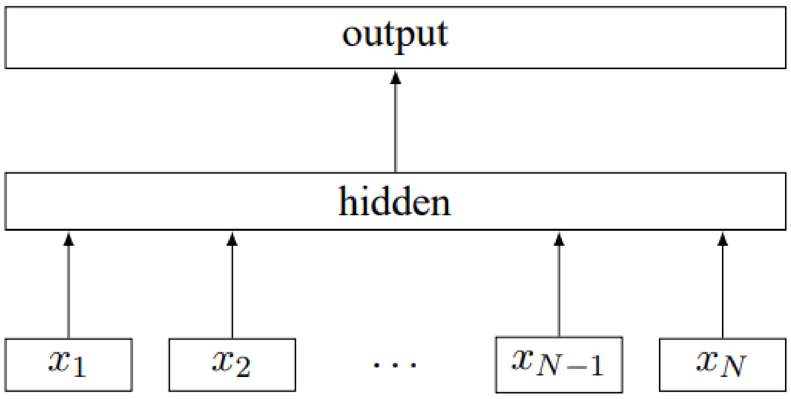

Figure 4.1: Internal structure of fasttext classifier[16].

In aspect category sentiment analysis, each sentence is assigned with one, or more aspect categories and corresponding sentiment labels. Therefore, it is a multi-label classification task. Aspect categories come from a predefined set and their sentiment polarities are positive, negative, or neutral. In this method, we concatenated aspect category labels with labels of sentiment polarities by underscore "_" char. Given set of L aspect categories $C = \{c_1...c_L\}$ and sentiment polarities $P = \{positive, negative, neutral\}$, the total number of all possible labels was determined as $all\_labels = \{c``\_"p : c \in C, p \in P\}$. Format of data was augmented for fasttext classifier, example is in 4.2.

$$\_\_label\_\_service\_negative\ \_\_label\_\_food\_positive\quad \begin{array}{l} the\ service\ was\ horrible \\ but\ the\ pizza\ was\ great \end{array} \tag{4.2}$$

Trained fasttext model was then used to classify samples from test set. In the case of binary classification, the output label was the one with the highest probability. For multi-label classification, we set a probability threshold that determines the set of tags we predict for certain training sentences.

4.2 Two-step fasttext classifier

As opposed to joint training and classification of aspect categories and their sentiment polarities, we came up with two-step fasttext classifier exclusively for ACSA. This method divides ACSA into two subtasks: ACD and SP. These subtasks are approached individually and performed one after another. Therefore, we do not have to concatenate aspect categories with their corresponding sentiment polarities. Ultimately, the detection of aspect categories is multi-label but subsequent sentiment polarity classification is multiclass.

Firstly, aspect category detection is performed. Given a set of L aspect categories $C = \{c_1 ... c_L\}$, each sentence is assigned with one, or more aspect categories. Obtained fasttext model from the training phase is then used to detect aspect categories in all sentences from test set. Detected aspect categories are later used in sentiment polarity classification.

Sentiment polarity classification makes use of detected aspect categories from the previous step. The fasttext model for sentiment polarity classification was trained on the same sentences as aspect category detection. For the purpose of the learning process, model is fed with sentences which have been linked together with original gold annotations. An example of such concatenation is shown in 4.3.

$$< text\ of\ sentence > [SEP] < aspect\ category > \tag{4.3}$$

If one sentence has multiple aspect categories, such sentence is added to input data multiple times joined by different aspect category in each instance. To evaluate this model, we concatenate sentences from test set with detected aspect categories from previous classification. Then, the model outputs predicted sentiment polarities. This concludes classification process of this method because we obtained predicted aspect categories and sentiment polarities. Complete overview of the proposed architecture is shown in Figure 4.2.

4.3 Classification using Logistic regression

Another method we propose is logistic regression. In this approach, we make use of logistic regression both for binary and multi-label classification. Logistic regression determines the probability of a sample being in a certain class by sigmoid function. If the number of possible classes is higher than two, we have to use multinomial logistic regression with softmax function. This is the alternative approach to joint fasttext classification. We also incorporated embedding framework USE[11] that can represent sentences as a whole, not just as an averaged sequence of words. It is optimized for greater-than-word length text.

Even though the ACSA task is multi-label, we came up with the method to train individual binary classifiers. We obtained labels by performing sentiment polarity classification after aspect category detection. Such structure of the classification pipeline split the initial multilabel task into multiple binary classifications thus allowing to use an ensemble of binary logistic regressions. Firstly, the input sentence samples were encoded by USE framework[11]. Then, we trained binary logistic regression for each aspect category. Individual classifiers determine whether the given sentence falls into a certain aspect category, or not. This is the first phase of our approach that handles aspect

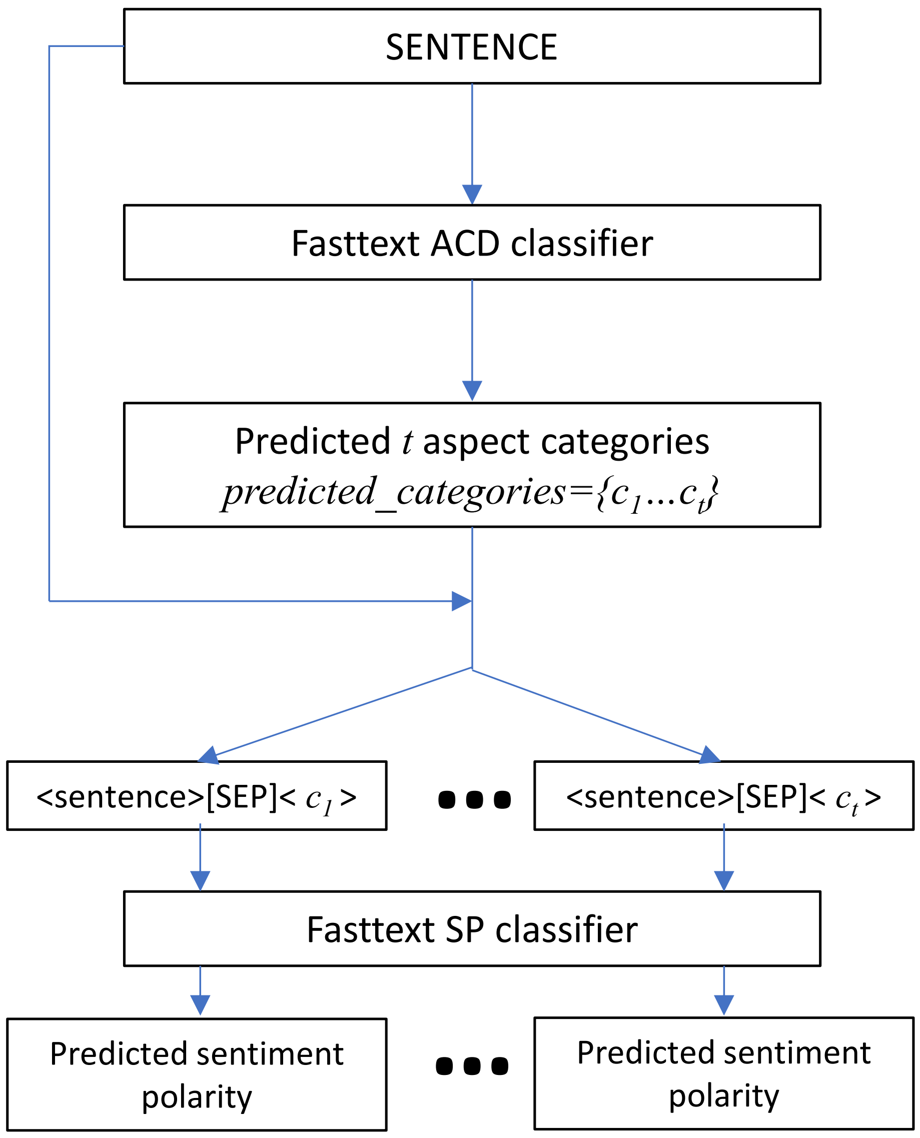
SENTENCE
Fasttext ACD classifier
Predicted t aspect categories
$predicted\_categories=\{c_1 \ldots c_t\}$
<sentence>[SEP]< c_1 >
<sentence>[SEP]< c_t >
Fasttext SP classifier
Predicted sentiment polarity
Predicted sentiment polarity

Figure 4.2: Architecture of two-step fasttext classifier.

category detection. To do sentiment polarity classification, we implemented another ensemble of logistic regressions. In this part of the classification, it is the ensemble of multinomial logistic regressions because sentiment is positive, negative, or neutral. The number of multinomial logistic regressions corresponds to the number of aspect categories. Each multinomial logistic regression classifies the sentiment polarity of samples from one particular aspect category. The output of this ensemble of multinomial logistic regressions is the sentiment polarity label for each aspect category detected in the previous step. This concludes the classification process of this method because we obtained predicted aspect categories and their corresponding sentiment polarity labels. Visualization of the model's architecture is shown in Figure 4.3.

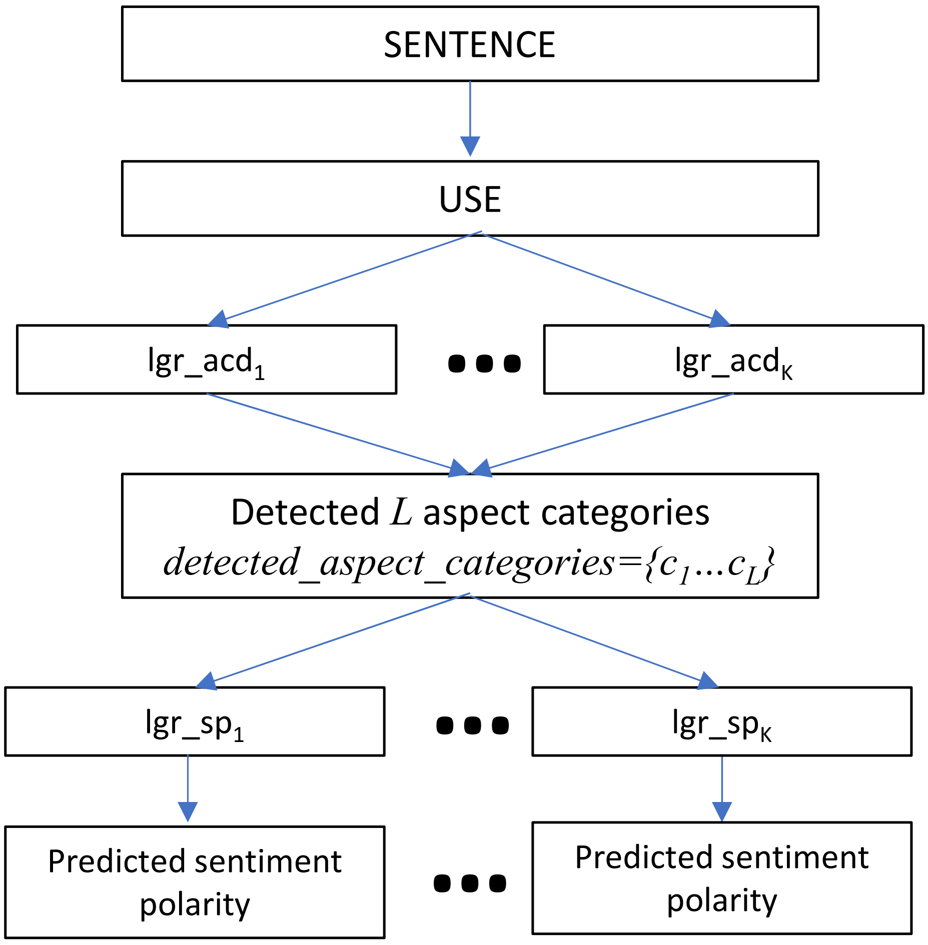
SENTENCE
USE
lgr_acd1
lgr_accK
Detected L aspect categories
detected_aspect_categories={c1...cL}
lgr_sp1
lgr_spK
Predicted sentiment polarity
Predicted sentiment polarity

Figure 4.3: Architecture of ensemble of logistic regressions.

Formally, given K aspect categories $C = \{c_1 ... c_K\}$, in the phase of aspect category detection, we construct the ensemble of K binary logistic regressions $LGR\_ACD = \{lgr\_acd_1 ... lgr\_acd_K\}$. To determine sentiment polarity, we implemented chain of multiple multinomial logistic regressions. The number of multinomial logistic regressions corresponds to the number of aspect categories. Let $LGR\_SP = \{lgr\_sp_1 ... lgr\_sp_K\}$ be the ensemble of multinomial logistic regressions. The outputs are sentiment polarities (positive, negative, or neutral) towards L detected aspect categories $detected\_aspect\_categories = \{c_1...c_L\}$ with respect to given input sentence.

Regarding sentence-level and document-level sentiment analysis, it is a binary classification problem. Therefore, we proposed the solution of using binary logistic regression because the sentiment polarity label for each sentence or block of text is either positive or negative. Using binary logistic regression, we predict the sentiment polarity of each sample from the test set. The input of such classifier is sentence or block of text encoded by USE[11] to embedding vector. The output is a sentiment polarity label for a given sample: positive, or negative.

Chapter 5

Experiments

In this chapter, we describe our experiments and discuss the results we have obtained. Since the nature of our datasets is ACSA, or sentence-level and document-level sentiment analysis, experiments on NER are not relevant so we do not provide such results in this chapter.

5.1 Used Datasets

5.1.1 Datasets for document-level sentiment analysis

For document-level sentiment analysis, we have used IMDB[44] dataset.

IMDB dataset[44] consists of highly polar movie reviews scrapped from IMDB[1]. They are labelled either as positive, or negative. Analysis of IMDB dataset is shown in Table 5.1.

	Train	Test	Dev
# samples	20000	25000	5000
avg # words per sample(with/without stopwords)	230/105	225/103	232/106
#positive	10000	12500	2500
#negative	10000	12500	2500

Table 5.1: Analysis of IMDB[44] dataset.

5.1.2 Datasets for sentence-level sentiment analysis

For sentence-level sentiment analysis, we have used Sentiment140[43] dataset.

Sentiment140 dataset[43] is a large-scale dataset composed of tweets. The annotation of each tweet is either positive, or negative. Table 5.2 shows the analysis of this dataset.

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

	Train	Test
# samples	1280000	320000
avg # words per sample(with/without stopwords)	13/7	13/7
#positive	640156	159844
#negative	639844	160156

Table 5.2: Analysis of Sentiment140[43] dataset.

5.1.3 Datasets for aspect category sentiment analysis

For the ACSA, we have used three datasets: res14[56], Restaurant large[59] and the ACSA variant of MAMS[60].

In the res14 dataset[56], English restaurant reviews (a subset of them come from [75]) were manually annotated. Aspect categories are from the predefined set of labels: *FOOD, SERVICE, PRICE, AMBIENCE, MISC.* Sentiment polarities towards given categories are *positive, negative, neutral.* Analysis of the res14 dataset is shown in 5.3.

	Train	Test
# samples	2898	767
avg # words per sample(with/without stopwords)	13.3/8	13.6/8.3
% of samples with more than one label	19	22
#positive	2164	728
#negative	807	196
#neutral	637	196
#FOOD	1166	402
#SERVICE	562	167
#PRICE	302	80
#AMBIENCE	385	105
#MISC	1103	219

Table 5.3: Analysis of res14[56] dataset.

MAMS dataset[60], its ACSA variant in particular, is composed of English reviews of restaurants. Aspect categories are: *FOOD, SERVICE, STAFF, PRICE, AMBIENCE, MENU, PLACE, MISCELLANEOUS.* Sentiment polarities towards them are: *positive, negative, neutral.* In this dataset, sentences consist of at least two aspect categories with different sentiment polarities. Table 5.4 shows the statistics and label distribution of MAMS dataset.

	Train	Test	Dev
# samples	3149	400	400
avg # words per sample(with/without stopwords)	23.4/17.5	23/17.3	23.1/17.3
% of samples with more than one label	100	100	100
#positive	1929	245	241
#negative	2084	263	259
#neutral	3077	393	388
#FOOD	2307	291	290
#SERVICE	631	78	84
#PRICE	322	38	45
#AMBIENCE	324	32	36
#MENU	475	76	51
#PLACE	694	81	88
#STAFF	1383	169	165
#MISCELLANEOUS	954	136	129

Table 5.4: Analysis of MAMS[60] dataset.

Restaurant large dataset[59] was created by merging datasets from restaurant domains published in [56, 57, 58]. Aspect categories include: *RESTAURANT, FOOD, DRINKS, AMBIENCE, SERVICE, PRICE, MISC and LOCATION*. Labels of sentiment polarity are *positive, negative and neutral*. Analysis of the Restaurant large dataset is in Table 5.5.

	Train	Test
# samples	3815	1963
avg # words per sample(with/without stopwords)	13.6/8	13.2/8.2
% of samples with more than one label	19	20
#positive	2710	1505
#negative	1198	680
#neutral	757	241
#FOOD	1597	928
#SERVICE	802	475
#PRICE	321	83
#AMBIENCE	525	243
#DRINKS	39	56
#RESTAURANT	245	386
#LOCATION	13	21
#MISC	1123	234

Table 5.5: Analysis of Restaurant large[59] dataset.

5.2 Evaluation metrics

There are several evaluation metrics that can be used to assess the performance of our classification. In this thesis, we use precision, recall, f1-score and accuracy. These evaluation metrics allow us to compare models with each other.

Therefore, we can determine the best architecture for our selected classification task. In our result section 5.4, we use micro average of mentioned metrics. In the case of ACSA task, the pair ($aspect\,category$, $sentiment\,polarity$) has to be completely correct to consider it as one of the correct predictions for the given sample. Formally, we define our evaluation metrics as follows:

- precision
$$\frac{T_p}{T_p + F_p}$$
- recall
$$\frac{T_p}{T_p + F_n}$$
- F1-score
$$2 \times \frac{Precision \times Recall}{Precision + Recall}$$
- accuracy
$$\frac{\#\, correct\, predictions}{\#\, all\, predictions}$$

T_p is number of true positives. F_n is number of false negatives whereas F_p is number of false positives.

5.3 Implementation details

Our experiments were developed using Python 3.9 programming language and PyCharm IDE. We trained and tested our methods on a personal notebook with *Intel(R) Core(TM) i5-9300H 2.40 GHz* CPU and 16GB of RAM. We performed a number of data preprocessing steps before training our classifiers. Such data preprocessing procedures include: lemmatization, removal of leading and trailing spaces, conversion of uppercase characters to lowercase and removal of HTML elements and punctuation. In the case of fasttext models, we also removed stopwords (using scikit-learn[2] and NLTK[3]) because such trained classifiers give better results. We measured training time using Python time module[4] and memory allocations using Python tracemalloc module[5].

Regarding fasttext classifier implementation, we used Python library fasttext[6] and its fasttext supervised classifier. For fine-tuning pretrained embeddings,

[2]https://scikit-learn.org/stable/modules/classes.html#module-sklearn.feature_extraction.text
[3]https://www.nltk.org/search.html?q=stopwords
[4]https://docs.python.org/3/library/time.html
[5]https://docs.python.org/3/library/tracemalloc.html
[6]https://pypi.org/project/fasttext/

we chose 300-dimensional English vectors[7] which were trained on Common Crawl and Wikipedia corpus. Since res14[56], Restaurant large[59] and Sentiment140[43] datasets do not provide official development split, we took 10% of all samples and chose it from a randomly shuffled train set. We used development sets to find the best combination of hyperparameters with an automatic hyperparameter optimization tool from fasttext library.

Methods based on logistic regression were implemented by scikit-learn library[8]. In addition, we used USE embeddings, both DAN-based[9] and Transformer-based[10] variants.

5.4 Results

We discuss our results in this section. In section 5.4.2, we discuss the results of sentence-level sentiment analysis and section 5.4.1 presents the results of document-level sentiment analysis. Section 5.4.3 presents the results of the aspect category detection subtask and section 5.4.4 presents the results of aspect category sentiment analysis.

5.4.1 Document-level sentiment analysis

In Table 5.6, we present results of document-level sentiment analysis evaluated on IMDB dataset[44]. The fasttext model with fine-tuning pre-trained embeddings has only slightly better accuracy than the fasttext model trained from scratch. It indicates that such pretrained embeddings do not hold crucial data needed for this kind of classification task. Both logistic regressions with DAN-based and Transformer-based USE embeddings give the same results in terms of accuracy. SOTA from [41] fine-tuned RoBERTA model[35] to perform this classification task.

Model	Acc	Training(s)	Memory(MB)
fasttext$_{\text{ours}}$	0.87	**43**	**139**
fasttext with pretrained vectors[76]$_{\text{ours}}$	**0.89**	286	140
lgr with USE (DAN variant)$_{\text{ours}}$	0.86	230	1069
lgr with USE (Transformer-based variant)$_{\text{ours}}$	0.86	520	1132
fine-tuned RoBERTA model[41]$_{\text{sota}}$	0.96		

Table 5.6: Results of document-level sentiment analysis on IMDB dataset[44].

In general, document-level sentiment analysis is a simpler task compared to aspect category sentiment analysis. Therefore, our proposed methods and

[7]`https://dl.fbaipublicfiles.com/fasttext/vectors-crawl/cc.en.300.bin.gz`
[8]`https://scikit-learn.org/stable/`
[9]`https://tfhub.dev/google/universal-sentence-encoder/4`
[10]`https://tfhub.dev/google/universal-sentence-encoder-large/5`

their results are comparable to SOTA models. In addition, our approaches are very time and memory efficient.

5.4.2 Sentence-level sentiment analysis

Table 5.7 shows results of sentence-level sentiment analysis on Sentiment140 dataset[43]. The fasttext model trained from scratch performs similarly as the fasttext classifier with fine-tuning pretrained embeddings. However, the fasttext model trained from scratch is three times faster. Logistic regression with Transformer-based USE embeddings has slightly better accuracy than its respective counterpart with DAN-based embeddings but training is significantly slower. SOTA from [42] proposed Maximum entropy classifier.

Model	Acc	Training(s)	Memory(MB)
fasttext$_{\text{ours}}$	0.77	**158**	**730**
fasttext with pretrained vectors[76]$_{\text{ours}}$	0.78	473	**730**
lgr with USE (DAN variant)$_{\text{ours}}$	0.78	477	2962
lgr with USE (Transformer-based variant)$_{\text{ours}}$	**0.81**	1350	3125
Maximum entropy classifier[42]$_{\text{sota}}$	0.88		

Table 5.7: Results of sentence-level sentiment analysis on Sentiment140 dataset[43].

On the whole, sentence-level sentiment analysis is a relatively simple task. Therefore, our time-efficient and less complex methods trained on CPU, achieve comparable results as state-of-the-art models. In addition, the Sentiment140 dataset has a large number of training samples. Combination of large-scale dataset and binary classification task results in the similar performance of fasttext models and logistic regressions with more sophisticated classifiers.

5.4.3 Aspect category detection

In Table 5.8, we present results of aspect category detection on MAMS dataset[60]. It shows that fasttext classification model outperforms logistic regression with USE embeddings by 1-4% in all presented evaluation metrics. Comparison of fasttext classifier with and without fine-tuning pretrained embeddings (300-dimensional English vectors from [76]) shows similar results. It indicates that pretrained vectors do not hold crucial contextual information needed for this classification. Also, training of fasttext classifier with fine-tuning pretrained embeddings is slower than training the fasttext classifier from scratch. Since the pretrained embedding file has 4.4GB, training process takes four minutes compared to the three-second learning process of model trained from scratch. Logistic regression with USE (DAN variant) has slightly worse performance than USE with Transformer-based embeddings. On the other hand, the training process of logistic regression with DAN-based USE embeddings is more than four times faster. SOTA result published in [53] shows better F1-score because it uses large language model BART[77].

Model	P	R	F1	Training(s)	Memory(MB)
fasttext$_{\text{ours}}$	**0.90**	0.81	**0.85**	**3**	**82**
fasttext with pretrained vectors[76]$_{\text{ours}}$	0.87	**0.83**	**0.85**	256	**82**
lgr with USE (DAN variant)$_{\text{ours}}$	0.81	0.81	0.81	12	133
lgr with USE (Transformer-based variant)$_{\text{ours}}$	0.83	0.82	0.82	52	196
BART model[53]$_{\text{sota}}$			0.91		

Table 5.8: Results of aspect category detection on MAMS dataset[60].

Table 5.9 shows results of ACD subtask on Restaurant large dataset[59]. Both fasttext models and logistic regressions with USE embeddings perform similarly in terms of F1-score. Fasttext with fine-tuning pretrained embeddings does not show significant improvement as opposed to the fasttext model trained from scratch. Therefore, it indicates that such pretrained embeddings do not provide crucial contextual or semantic information for the ACD subtask on this dataset. Logistic regression with DAN-based USE embeddings has lower F1-score and recall compared to the same model but with Transformer-based USE embeddings. Even though the fasttext model without fine-tuning pretrained embeddings and logistic regression with DAN-based USE embeddings achieve slightly worse results in terms of F1-scores, their training is faster. The learning process of such methods takes seconds as opposed to the training phases of their respective counterparts. SOTA from [54] uses model based on double BiLSTM.

Model	P	R	F1	Training(s)	Memory(MB)
fasttext$_{\text{ours}}$	0.65	**0.76**	0.70	**4**	**78**
fasttext with pretrained vectors[76]$_{\text{ours}}$	0.67	**0.76**	**0.71**	235	**78**
lgr with USE (DAN variant)$_{\text{ours}}$	**0.75**	0.65	0.69	13	145
lgr with USE (Transformer-based variant)$_{\text{ours}}$	0.70	0.72	**0.71**	58	208
double BiLSTM model[54]$_{\text{sota}}$			0.76		

Table 5.9: Results of aspect category detection on Restaurant large dataset[59].

Another set of ACD results is shown in Table 5.10. Logistic regression with Transformer-based USE outperforms all of our methods in terms of F1-score, precision and recall. Since res14 dataset[56] has the smallest number of training samples compared to the other datasets, more complex embedding frameworks and classification methods outperform fasttext models by a significant margin. Regarding fasttext models, we can see that classification with, or without fine-tuning pretrained vectors, give very similar results. Therefore, we consider that pretrained embeddings do not provide the necessary information for the ACD task. In addition, training of fasttext model with fine-tuning pretrained embeddings takes significantly longer. SOTA model from [53] approaches ACD task with BART[77].

Model	P	R	F1	Training(s)	Memory(MB)
fasttext$_{\text{ours}}$	0.76	0.83	0.80	**4**	**85**
fasttext with pretrained vectors[76]$_{\text{ours}}$	0.79	0.80	0.80	267	**85**
lgr with USE (DAN variant)$_{\text{ours}}$	0.83	0.87	0.85	13	119
lgr with USE (Transformer-based variant)$_{\text{ours}}$	**0.85**	**0.88**	**0.87**	48	182
BART model[53]$_{\text{sota}}$			0.93		

Table 5.10: Results of aspect category detection on res14 dataset[56].

Overall, aspect category detection is a simpler task. Therefore, our time-efficient, memory-efficient and less complex methods give slightly worse results than SOTA models that use large language models or deep learning techniques. However, our proposed methods can be trained on personal computers only with CPUs because they are memory efficient. Only results on res14 dataset show a larger margin between logistic regression and fasttext. The reason is that the res14 dataset has less training samples compared to other datasets. As a result, simpler encoding algorithms and classifiers give a worse performance.

5.4.4 Aspect category sentiment analysis

Results of ACSA task on MAMS dataset are shown in Table 5.11. In this case, logistic regression with Transformer-based USE embeddings outperforms all of our other methods with respect to F1-score and recall. Logistic regression with DAN-based USE embeddings has worse results but training is almost seven times faster compared to its respective counterpart. Two-step fasttext gives better results than one-step fasttext in terms of F1-score and recall. Also, our two-step fasttext architecture shows the best results being trained from scratch as opposed to one-step fasttext giving better results with fine-tuning pretrained embeddings. Therefore, the training process of two-step fasttext is significantly faster than the training of one-step fasttext. SOTA from [53] proposed architecture based on BART[77].

Model	P	R	F1	Training(s)	Memory(MB)
one-step fasttext$_{\text{ours}}$	**0.61**	0.45	0.52	297	**82**
two-step fasttext$_{\text{ours}}$	0.58	0.51	0.54	**4**	**82**
lgr with USE (DAN variant)$_{\text{ours}}$	0.57	0.57	0.57	28	133
lgr with USE (Transformer-based variant)$_{\text{ours}}$	0.59	**0.59**	**0.59**	188	196
BART model[53]$_{\text{sota}}$			0.77		

Table 5.11: Results of ACSA on MAMS dataset[60].

Table 5.12 shows the results of ACSA on Restaurant large dataset[59]. In terms of F1-score, logistic regression with Transformer-based USE embeddings gives better results than the second variant of the logistic regression model. Compared to its respective counterpart with DAN-based USE embeddings, the training process is significantly slower. The one-step fasttext model, with fine-tuning pretrained embeddings, has worse results of recall and F1-score than the two-step fasttext classifier, also with fine-tuning pretrained embeddings. However, training of one-step fasttext is two times faster. SOTA from [54] uses BERT[33] to approach the ACSA task on this dataset.

Model	P	R	F1	Training(s)	Memory(MB)
one-step fasttext$_{\text{ours}}$	0.58	0.58	0.58	228	**79**
two-step fasttext$_{\text{ours}}$	0.58	**0.61**	**0.60**	516	82
lgr with USE (DAN variant)$_{\text{ours}}$	0.56	0.57	0.57	**20**	144
lgr with USE (Transformer-based variant)$_{\text{ours}}$	**0.60**	**0.61**	**0.60**	178	208
BERT model[54]$_{\text{sota}}$			0.77		

Table 5.12: Results of ACSA on Restaurant large dataset[59].

Table 5.13 shows results of the ACSA task on res14 dataset[56]. Both variants of fasttext classifier incorporate fine-tuning of pretrained embeddings. The two-step fasttext model outperforms the one-step fasttext by 3% and 5% in terms of F1 and recall. However, training of one-step fasttext model is two times faster. Logistic regression with DAN-based embeddings has better overall results than both fasttext classifiers. Yet, the best results come from logistic regression with Transformer-based USE embeddings. It beats its respective counterparts in all evaluation metrics but training is significantly slower than in the case of logistic regression with DAN-based USE embeddings. SOTA from [54] uses BERT[33] to approach the ACSA task on this dataset.

Model	P	R	F1	Training(s)	Memory(MB)
one-step fasttext$_{\text{ours}}$	0.60	0.56	0.58	235	**85**
two-step fasttext$_{\text{ours}}$	0.60	0.61	0.61	498	**85**
lgr with USE (DAN variant)$_{\text{ours}}$	0.69	0.72	0.70	**14**	119
lgr with USE (Transformer-based variant)$_{\text{ours}}$	**0.74**	**0.76**	**0.75**	103	182
BERT model[54]$_{\text{sota}}$			0.85		

Table 5.13: Results of ACSA on res14 dataset[56].

On the whole, our proposed methods are time-efficient, memory-efficient and can be trained on the CPU of the personal computer. Compared to ACD, ACSA is more complex task that requires a better understanding of context when classifying sentiment polarities with respect to detected aspect categories. Therefore, the margins between our results and SOTA results are wider than in the case of the ACD task. Regarding our proposed methods, logistic regression with Transformer-based USE outperforms other models on MAMS dataset in terms of recall and F1-score. The reason is that each sentence in this dataset is labelled with at least two aspect categories with different polarities. Thus, the more advanced embedding framework is required. Similarly, on the res14 dataset, logistic regression with Transformer-based USE has the best results. It is because the res14 dataset has fewer training samples compared to other datasets. So, simpler encoding algorithms and classificators have worse results.

Chapter 6

Conclusion

In this thesis, we have introduced and described the problems of sentiment analysis and named entity recognition. In the case of sentiment analysis, we focused on document-level sentiment analysis, sentence-level sentiment analysis, and aspect-based sentiment analysis, aspect category sentiment analysis in particular. We reviewed state-of-the-art methods and datasets currently used to approach these tasks. We also discussed the theoretical background of such methods. Chapter 2 explained concepts of word representations, traditional machine learning models, and neural networks as well as machine learning and natural language processing in general. The primary focus of our experiments was to create classifiers that can be trained on the CPU and have low computational requirements. Therefore, we proposed classification approaches based on fasttext supervised classifier and logistic regression combined with embedding framework named Universal Sentence Encoder. We have performed several experiments and we report results of sentence-level and document-level sentiment analysis, aspect category detection, and aspect category sentiment analysis in Chapter 5. Even though our models have achieved worse results compared to state-of-the-art, the common advantage of our approaches is low time and space complexity. As opposed to widely used deep learning techniques and large language models with millions of parameters that need GPUs for training, our proposed methods, with less complex internal architectures, are time and memory efficient.

Appendix A

Bibliography

[1] A. L. Samuel, "Some studies in machine learning using the game of checkers," *IBM Journal of Research and Development*, vol. 3, no. 3, pp. 210–229, 1959.

[2] JavaTpoint, "Applications of machine learning," accessed: 2023-04-03. [Online]. Available: https://www.javatpoint.com/applications-of-machine-learning

[3] Z. Zhou and S. Liu, *Machine Learning.* Springer Nature Singapore, 2021. [Online]. Available: https://books.google.cz/books?id=ctM-EAAAQBAJ

[4] Precedence Research. (2022) Natural language processing market size, report 2030. Accessed: 2023-05-13. [Online]. Available: https://www.precedenceresearch.com/natural-language-processing-market

[5] G. A. Miller, R. Beckwith, C. D. Fellbaum, D. Gross, and K. J. Miller, "Introduction to wordnet: An on-line lexical database," *International Journal of Lexicography*, vol. 3, pp. 235–244, 1990.

[6] P. S. G. Britain) and J. Firth, *Studies in Linguistic Analysis*, ser. Publications of the Philological Society. Blackwell, 1957. [Online]. Available: https://books.google.cz/books?id=JWktAAAAMAAJ

[7] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," *arXiv preprint arXiv:1607.04606*, 2016.

[8] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[9] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in Neural Information Processing Systems*, C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and

K. Weinberger, Eds., vol. 26. Curran Associates, Inc., 2013. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2013/file/9aa42b31882ec039965f3c4923ce901b-Paper.pdf

[10] J. Pennington, R. Socher, and C. D. Manning, “Glove: Global vectors for word representation,” in *Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1532–1543. [Online]. Available: http://www.aclweb.org/anthology/D14-1162

[11] D. Cer, Y. Yang, S. yi Kong, N. Hua, N. L. U. Limtiaco, R. S. John, N. Constant, M. Guajardo-Céspedes, S. Yuan, C. Tar, Y. hsuan Sung, B. Strope, and R. Kurzweil, “Universal sentence encoder,” in *In submission to: EMNLP demonstration*, Brussels, Belgium, 2018, in submission. [Online]. Available: https://arxiv.org/abs/1803.11175

[12] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. 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)*. New Orleans, Louisiana: Association for Computational Linguistics, Jun. 2018, pp. 2227–2237. [Online]. Available: https://aclanthology.org/N18-1202

[13] D. A. Huffman, “A method for the construction of minimum-redundancy codes,” *Proceedings of the IRE*, vol. 40, no. 9, pp. 1098–1101, 1952.

[14] J. Wieting, M. Bansal, K. Gimpel, and K. Livescu, “Charagram: Embedding words and sentences via character n-grams,” *CoRR*, vol. abs/1607.02789, 2016. [Online]. Available: http://arxiv.org/abs/1607.02789

[15] T. Kocmi and O. Bojar, “Subgram: Extending skip-gram word representation with substrings,” in *Text, Speech, and Dialogue*, P. Sojka, A. Horák, I. Kopeček, and K. Pala, Eds. Cham: Springer International Publishing, 2016, pp. 182–189.

[16] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov, “Bag of tricks for efficient text classification,” *arXiv preprint arXiv:1607.01759*, 2016.

[17] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, “Tensorflow: a system for large-scale machine learning.” in *Osdi*, vol. 16, no. 2016. Savannah, GA, USA, 2016, pp. 265–283.

[18] S. R. Bowman, G. Angeli, C. Potts, and C. D. Manning, “A large annotated corpus for learning natural language inference,” in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal: Association for Computational Linguistics, Sep. 2015, pp. 632–642. [Online]. Available: https://aclanthology.org/D15-1075

[19] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2017. [Online]. Available: https://arxiv.org/pdf/1706.03762.pdf

[20] M. Iyyer, V. Manjunatha, J. Boyd-Graber, and H. Daumé III, "Deep unordered composition rivals syntactic methods for text 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)*. Beijing, China: Association for Computational Linguistics, Jul. 2015, pp. 1681–1691. [Online]. Available: https://aclanthology.org/P15-1162

[21] D. Jurafsky and J. Martin, *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*, ser. Prentice Hall series in artificial intelligence. Pearson Prentice Hall, 2009. [Online]. Available: https://books.google.cz/books?id=fZmj5UNK8AQC

[22] M. Franzese and A. Iuliano, "Hidden markov models," in *Encyclopedia of Bioinformatics and Computational Biology*, S. Ranganathan, M. Gribskov, K. Nakai, and C. Schönbach, Eds. Oxford: Academic Press, 2019, pp. 753–762. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780128096338204883

[23] H. M. Wallach, "Conditional random fields: An introduction," *Technical Reports (CIS)*, p. 22, 2004.

[24] P. Král, "Features for named entity recognition in czech language," October 2011. [Online]. Available: https://www.researchgate.net/publication/256605620_Features_for_named_entity_recognition_in_Czech_language

[25] A. Krenker, J. Bešter, and A. Kos, "Introduction to the artificial neural networks," *Artificial Neural Networks: Methodological Advances and Biomedical Applications. InTech*, pp. 1–18, 2011.

[26] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," 1986.

[27] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, no. 5, pp. 359–366, 1989. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0893608089900208

[28] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[29] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *Journal of Machine Learning Research*, vol. 12, no. 61, pp. 2121–2159, 2011. [Online]. Available: http://jmlr.org/papers/v12/duchi11a.html

[30] Q. Yuan, Y. Dai, and G. Li, "Exploration of english speech translation recognition based on the lstm rnn algorithm," *Neural Computing and Applications*, pp. 1–10, 03 2023.

[31] G. Weiss, Y. Goldberg, and E. Yahav, "On the practical computational power of finite precision rnns for language recognition," *arXiv preprint arXiv:1805.04908*, 2018.

[32] T. Dozat and C. D. Manning, "Deep biaffine attention for neural dependency parsing," *arXiv preprint arXiv:1611.01734*, 2016.

[33] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[34] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[35] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," *arXiv preprint arXiv:1907.11692*, 2019.

[36] N. Barney, "Sentiment analysis (opinion mining)," accessed: 2023-04-07. [Online]. Available: https://www.techtarget.com/searchbusinessanalytics/definition/opinion-mining-sentiment-mining

[37] W. Medhat, A. Hassan, and H. Korashy, "Sentiment analysis algorithms and applications: A survey," *Ain Shams Engineering Journal*, vol. 5, no. 4, pp. 1093–1113, 2014. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2090447914000550

[38] L. Zhang, S. Wang, and B. Liu, "Deep learning for sentiment analysis: A survey," *WIREs Data Mining and Knowledge Discovery*, vol. 8, no. 4, p. e1253, 2018. [Online]. Available: https://wires.onlinelibrary.wiley.com/doi/abs/10.1002/widm.1253

[39] A. Iqbal, R. Amin, J. Iqbal, R. Alroobaea, A. Binmahfoudh, and M. Hussain, "Sentiment analysis of consumer reviews using deep learning," *Sustainability*, vol. 14, no. 17, p. 10844, 2022.

[40] N. Ali, M. Hamid, and A. Youssif, "Sentiment analysis for movies reviews dataset using deep learning models," *International Journal of Data Mining Knowledge Management Process*, vol. 09, pp. 19–27, 05 2019.

[41] Z. Bingyu and N. Arefyev, "The document vectors using cosine similarity revisited," in *Proceedings of the Third Workshop on Insights from Negative Results in NLP*. Dublin, Ireland: Association for Computational Linguistics, May 2022, pp. 129–133. [Online]. Available: https://aclanthology.org/2022.insights-1.17

[42] N. Iqbal, A. M. Chowdhury, and T. Ahsan, “Enhancing the performance of sentiment analysis by using different feature combinations,” in *2018 International Conference on Computer, Communication, Chemical, Material and Electronic Engineering (IC4ME2)*, 2018, pp. 1–4.

[43] A. Go, R. Bhayani, and L. Huang, “Twitter sentiment classification using distant supervision,” *CS224N project report, Stanford*, vol. 1, no. 12, p. 2009, 2009.

[44] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, “Learning word vectors for sentiment analysis,” in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies.* Portland, Oregon, USA: Association for Computational Linguistics, June 2011, pp. 142–150. [Online]. Available: http://www.aclweb.org/anthology/P11-1015

[45] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Ng, and C. Potts, “Recursive deep models for semantic compositionality over a sentiment treebank,” in *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing.* Seattle, Washington, USA: Association for Computational Linguistics, Oct. 2013, pp. 1631–1642. [Online]. Available: https://aclanthology.org/D13-1170

[46] P. Nakov, S. Rosenthal, Z. Kozareva, V. Stoyanov, A. Ritter, and T. Wilson, “SemEval-2013 task 2: Sentiment analysis in Twitter,” in *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013).* Atlanta, Georgia, USA: Association for Computational Linguistics, Jun. 2013, pp. 312–320. [Online]. Available: https://aclanthology.org/S13-2052

[47] B. Liu, “Sentiment analysis and opinion mining,” *Synthesis lectures on human language technologies*, vol. 5, no. 1, pp. 1–167, 2012.

[48] W. Zhang, X. Li, Y. Deng, L. Bing, and W. Lam, “A survey on aspect-based sentiment analysis: Tasks, methods, and challenges,” *IEEE Transactions on Knowledge and Data Engineering*, pp. 1–20, 2022.

[49] H. H. Do, P. Prasad, A. Maag, and A. Alsadoon, “Deep learning for aspect-based sentiment analysis: A comparative review,” *Expert Systems with Applications*, vol. 118, pp. 272–299, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417418306456

[50] H. Cai, Y. Tu, X. Zhou, J. Yu, and R. Xia, “Aspect-category based sentiment analysis with hierarchical graph convolutional network,” in *Proceedings of the 28th International Conference on Computational Linguistics.* Barcelona, Spain (Online): International Committee on Computational Linguistics, Dec. 2020, pp. 833–843. [Online]. Available: https://aclanthology.org/2020.coling-main.72

[51] M. Schmitt, S. Steinheber, K. Schreiber, and B. Roth, "Joint aspect and polarity classification for aspect-based sentiment analysis with end-to-end neural networks," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing.* Brussels, Belgium: Association for Computational Linguistics, Oct.-Nov. 2018, pp. 1109–1114. [Online]. Available: https://aclanthology.org/D18-1139

[52] Z. Zeng, J. Ma, M. Chen, and X. Li, "Joint learning for aspect category detection and sentiment analysis in chinese reviews," in *Information Retrieval*, Q. Zhang, X. Liao, and Z. Ren, Eds. Cham: Springer International Publishing, 2019, pp. 108–120.

[53] J. Liu, Z. Teng, L. Cui, H. Liu, and Y. Zhang, "Solving aspect category sentiment analysis as a text generation task," in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing.* Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 4406–4416. [Online]. Available: https://aclanthology.org/2021.emnlp-main.361

[54] Y. Fu, J. Liao, Y. Li, S. Wang, D. Li, and X. Li, "Multiple perspective attention based on double bilstm for aspect and sentiment pair extract," *Neurocomputing*, vol. 438, pp. 302–311, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231221001557

[55] Y. Wang, A. Sun, M. Huang, and X. Zhu, "Aspect-level sentiment analysis using as-capsules," in *The World Wide Web Conference*, ser. WWW '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 2033–2044. [Online]. Available: https://doi.org/10.1145/3308558.3313750

[56] M. Pontiki, D. Galanis, J. Pavlopoulos, H. Papageorgiou, I. Androutsopoulos, and S. Manandhar, "SemEval-2014 task 4: Aspect based sentiment analysis," in *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014).* Dublin, Ireland: Association for Computational Linguistics, Aug. 2014, pp. 27–35. [Online]. Available: https://aclanthology.org/S14-2004

[57] M. Pontiki, D. Galanis, H. Papageorgiou, S. Manandhar, and I. Androutsopoulos, "SemEval-2015 task 12: Aspect based sentiment analysis," in *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015).* Denver, Colorado: Association for Computational Linguistics, Jun. 2015, pp. 486–495. [Online]. Available: https://aclanthology.org/S15-2082

[58] M. Pontiki, D. Galanis, H. Papageorgiou, I. Androutsopoulos, S. Manandhar, M. AL-Smadi, M. Al-Ayyoub, Y. Zhao, B. Qin, O. De Clercq, V. Hoste, M. Apidianaki, X. Tannier, N. Loukachevitch, E. Kotelnikov, N. Bel, S. M. Jiménez-Zafra, and G. Eryiğit, "SemEval-2016 task 5: Aspect based sentiment analysis," in *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016).*

San Diego, California: Association for Computational Linguistics, Jun. 2016, pp. 19–30. [Online]. Available: https://aclanthology.org/S16-1002

[59] W. Xue and T. Li, "Aspect based sentiment analysis with gated convolutional networks," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Melbourne, Australia: Association for Computational Linguistics, Jul. 2018, pp. 2514–2523. [Online]. Available: https://aclanthology.org/P18-1234

[60] Q. Jiang, L. Chen, R. Xu, X. Ao, and M. Yang, "A challenge dataset and effective models for aspect-based sentiment analysis," 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)*. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 6280–6285. [Online]. Available: https://aclanthology.org/D19-1654

[61] S. Tu, *Experiments on Approaches to Named Entity Recognition in IsiZulu*. Georgetown University, 2021. [Online]. Available: http://hdl.handle.net/10822/1062365

[62] S. Zheng, "Sequence labeling." [Online]. Available: https://stevezheng23.github.io/sequence_labeling_tf/

[63] U. Naseem, M. Khushi, V. Reddy, S. Rajendran, I. Razzak, and J. Kim, "Bioalbert: A simple and effective pre-trained language model for biomedical named entity recognition," in *2021 International Joint Conference on Neural Networks (IJCNN)*, 2021, pp. 1–7.

[64] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, "Albert: A lite bert for self-supervised learning of language representations," *arXiv preprint arXiv:1909.11942*, 2019.

[65] X. Ma and E. Hovy, "End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 1064–1074. [Online]. Available: https://aclanthology.org/P16-1101

[66] X. Li, X. Sun, Y. Meng, J. Liang, F. Wu, and J. Li, "Dice loss for data-imbalanced NLP tasks," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, Jul. 2020, pp. 465–476. [Online]. Available: https://aclanthology.org/2020.acl-main.45

[67] X. Li, J. Feng, Y. Meng, Q. Han, F. Wu, and J. Li, "A unified mrc framework for named entity recognition," *arXiv preprint arXiv:1910.11476*, 2019.

[68] Z. Yang, R. Salakhutdinov, and W. Cohen, "Multi-task cross-lingual sequence tagging from scratch," *arXiv preprint arXiv:1603.06270*, 2016.

[69] M. Straka, J. Straková, and J. Hajič, "Czech text processing with contextual embeddings: Pos tagging, lemmatization, parsing and ner," in *Text, Speech, and Dialogue: 22nd International Conference, TSD 2019, Ljubljana, Slovenia, September 11–13, 2019, Proceedings 22*. Springer, 2019, pp. 137–150.

[70] E. F. Tjong Kim Sang and F. De Meulder, "Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition," in *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, 2003, pp. 142–147. [Online]. Available: https://aclanthology.org/W03-0419

[71] R. Weischedel, M. Palmer, M. Marcus, E. Hovy, S. Pradhan, L. Ramshaw, N. Xue, A. Taylor, J. Kaufman, M. Franchini, M. El-Bachouti, R. Belvin, and A. Houston, "OntoNotes Release 5.0," 2013. [Online]. Available: https://hdl.handle.net/11272.1/AB2/MKJJ2R

[72] M. Ševčíková, Z. Žabokrtský, and O. Krůza, "Named entities in czech: Annotating data and developing NE tagger," in *Lecture Notes in Artificial Intelligence, Proceedings of the 10th International Conference on Text, Speech and Dialogue*, ser. Lecture Notes in Computer Science, V. Matoušek and P. Mautner, Eds., vol. 4629, no. XVII. Berlin / Heidelberg: Springer, 2007, pp. 188–195.

[73] A. Ghaddar and P. Langlais, "Winer: A wikipedia annotated corpus for named entity recognition," in *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, 2017. [Online]. Available: http://www.aclweb.org/anthology/I17-1042

[74] E. Faessler, L. Modersohn, C. Lohr, and U. Hahn, "ProGene - a large-scale, high-quality protein-gene annotated benchmark corpus," in *Proceedings of the Twelfth Language Resources and Evaluation Conference*. Marseille, France: European Language Resources Association, May 2020, pp. 4585–4596. [Online]. Available: https://aclanthology.org/2020.lrec-1.564

[75] G. Ganu, N. Elhadad, and A. Marian, "Beyond the stars: Improving rating predictions using review text content," in *International Workshop on the Web and Databases*, 2009.

[76] E. Grave, P. Bojanowski, P. Gupta, A. Joulin, and T. Mikolov, "Learning word vectors for 157 languages," in *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*, 2018.

[77] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer, "BART: Denoising sequence-to-sequence pre-training for natural language generation,

translation, and comprehension," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, Jul. 2020, pp. 7871–7880. [Online]. Available: https://aclanthology.org/2020.acl-main.703

Appendix B

List of shortcuts

ABSA: Aspect-based sentiment analysis
ACD: Aspect category detection
ACSA: Aspect category sentiment analysis
AS-Capsules: Aspect-level sentiment capsules
Acc: Accuracy
AdaGrad: Adaptive gradient algorithm
Adam: Adaptive Moment Estimation
BART: Bidirectional Auto-Regressive Transformers
BERT: Bidirectional Encoder Representations from Transformers
BiLSTM: Bidirectional Long short-term memory
CBOW: Continuous Bag of Words
CNN: Convolutional neural network
CRF: Conditional random field
DAN: Deep averaging network
F1: F1-score
GCN: Graph Convolutional Network
GPT-3: third generation Generative Pre-trained Transformer
GRU: Gated recurrent unit
HMM: Hidden Markov model
Hier-GCN: Hierarchical Graph Convolutional Network
IDE: Integrated development environment
IOB: Inside-Outside-Beginning
LSTM: Long short-term memory
ML: Machine Learning
MLP: Multilayer perceptron
NER: Named entity recognition
NLP: Natural language processing
NLTK: Natural Language Toolkit
NN: Neural network
OTE: Opinion term extraction
P: Precision
POS: Part-of-speech tagging

R: Recall
RNN: Recurrent neural network
RoBERTa: A Robustly Optimized BERT Pretraining Approach
SA: Sentiment analysis
SG: Skip-gram
SGD: Stochastic gradient descent
SOTA: State-of-the-art
SP: Sentiment polarity
USE: Universal Sentence Encoder
lgr: Logistic regression