University of Novi Sad
Faculty of Sciences
Department of Mathematics
and Informatics

Analysis of Machine Learning and Deep Learning Algorithms for Text Sentiment Detection

Master Thesis

Stefan Dimitrijević

Supervisor: dr Miloš Savić

Novi Sad, 2023

Preface

Abstract

Sentiment analysis or opinion mining is the task of automatically extracting and classifying the sentiment of the text. It can be applied in numerous fields such as marketing, customer service, etc.

Sentiment analysis has gained much attention in recent years. The goal of this thesis is to make a comparison of machine learning and deep learning models. Such a comparison is key for understanding the potential limitations, advantages, and disadvantages of popular methods.

In this thesis, we focus on three datasets: movie reviews (IMDB dataset), coronavirus tweets (Corona dataset), and product and service reviews (YELP reviews).

In chapter 1, we introduce the problem, and in the following chapter 2, we go through the literature that tries to solve the sentiment analysis problem. In chapter 3, we give an explanation of each dataset individually, whereas, in chapter 4, we give dive into feature engineering from textual data. In chapter 5 we go briefly through some machine learning concepts. Chapters 6 and 7 consist of machine learning and deep learning models, respectively. In chapter 8 we talk about the optimization of neural networks. In chapter 9 we go through experiments and results achieved. In the last chapter, chapter 10, we draw conclusions.

Acknowledgements

I would like to express my deepest and most sincere gratitude to my thesis supervisor dr. Miloš Savić for his motivation and support during research on this topic.

Finally, I must express my profound gratitude to my parents and my family for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them. Thank you.

Contents

List of Figures

List of Tables

List of Abbreviations

DT	- Decision Tree
RF	- Random Forest
SVM	- Support Vector Machine
LR	- Logistic Regression
NB	- Naive Bayes
MLP	- Multi-layer Perceptron
RNN	- Recurrent Neural Network
LSTM	- Long Short-Term Memory
BERT	- Bidirectional Encoder Representations from Transformers
SGD	- Stochastic Gradient Descent
BOW	- Bag of Words
TF-IDF	- Term Frequency Inverse Document Frequency

Chapter 1

Introduction

Sentiment analysis also referred to as opinion mining, is the task of automatically extracting and classifying the sentiment of text [1]. It can be used in different fields such as marketing [2], customer service [3], social media [4], etc. Getting accurate sentiment prediction of text can help businesses understand their customers, opinions, and emotions, allowing them to make strategic planning based on this information [5].

Sentiment analysis is a part of Natural Language Processing (NLP). Early approaches to sentiment analysis were not sophisticated as they are today. Mostly, they were based on lexicon-based methods, where dictionaries of words and phrases were created manually with positive and negative sentiments [6]. This is a very simple approach and easy to implement but it had its limitations in terms of accuracy and generalization to new domains and languages [1].

These algorithms used a variety of features, such as word n-grams [7], part-of-speech tags [8], and syntactic parsing [9], to learn the relationship between text and sentiment. These techniques demonstrated greater accuracy compared to lexicon-based methods they still had certain limitations, such as they could not grasp the complex relationship between words and contextual information.

Recently, deep learning started to emerge as a very strong tool for sentiment analysis [10]. Deep learning models such as convolution neural networks (CNN) [11] and recurrent neural networks (RNN) [10], have achieved state-of-the-art in sentiment analysis, as well as in other NLP tasks. These deep learning models can capture complex patterns in text data [12]. However, they also have their limitation, such as requiring large computational resources and huge amounts of training data [12], and it is difficult to explain their decision-making process [13].

A few years ago, completely new architectural models emerged using so-called attention which dealt with long-term dependencies and complex propagation through time [14]. Based on the attention architecture, many models were developed for NLP tasks, including BERT [15].

In this thesis, we present a comprehensive study of sentiment analysis, the current state of the art, and future directions. We review the various techniques and algorithms that have been developed for this task and discuss the challenges and limitations of these methods. We also present our contributions to the field of sentiment analysis and suggest directions for future research.

Chapter 2

Related Work

In this section, we review sentiment analysis, including the various techniques and algorithms that have been developed for this task. Sentiment analysis started to gain more attention and research at the beginning of the 2000s.

The initial approach to solving this problem is to apply lexicon-based methods. There was a lot of feature crafting and all this led to overall accuracy between 50% and 70%, depending on the dataset [16]. For sentiment analysis in blogs, authors in [17] have done a comparison between the lexicon-based method and Naive Bayes (NB), which showed that lexicon-based methods seriously underperform in comparison to Naive Bayes, accuracy can be 20% lower. Some authors tried even a rule-based [18] approach which only lead to 55% accuracy.

At the end of the first decade in 2000, researchers started using more statistical algorithms for sentiment analysis. Researchers and practitioners also started comparing different statistical models. Authors in [19] have done a comparison of Naive Bayes and Support Vector Machine (SVM), which lead to NB outperforming SVM by more than 10%. In study [20], authors tried SVM, NB, Random Forest (RF), and Logistic Regression (LR). The final results showed that there was less than a 1% difference between different algorithms.

All previous approaches treated the text as a bag of words where the order is not important. Word order can have a large impact on the overall meaning of the sentence. That is where word embeddings come from. They tackle the issue of high dimensionality and word order. The first word embedding was Word2Vec [21], followed by GloVe [22], and FastText [23]. They are trained using neural networks. At the same time, other deep neural networks become popular, such as CNN and RNN, which have promised good results in classification tasks. Authors in [24] have done benchmarks of Word2Vec and GloVe and also used refinement techniques for better performance in sentiment analysis. GloVe gave a better performance than Word2Vec in combination with CNN and Long-Short Term Memory, one variant of RNN, giving a few percentages of accuracy. Authors in [10] also showed that CNN with GloVe achieved 2% and 4% greater accuracy than LR and NB with bag-of-words, respectively. They also showed that BERT outperformed Glove in combination with CNN. Authors in [25] showed that a combination of BERT embeddings with CNN and LSTM consistently outperformed Word2Vec and GloVe with CNN and LSTM.

Chapter 3

Datasets

For sentiment analysis, three different datasets are used movie reviews (referred to as the IMDB dataset), tweets about the coronavirus, and reviews about products and services (referred to as the YELP dataset).

1. IMDB reviews
   IMDB dataset[1] contains 25k movie reviews in CSV format. Labels for movies are:

   (a) Positive

   (b) Negative

   Data distribution is balanced, with 12500 examples of both positive and negative classes, as shown in Figure 3.1.

2. YELP reviews
   The original YELP dataset[2] contains 650k yelp reviews. In order to have a similar number of instances across different datasets, and to make training possible, only 40k instances are sampled from the original dataset. It has five labels:

   (a) Extremely Negative

   (b) Negative

[1]Link towards IMDB dataset `https://www.kaggle.com/datasets/lakshmi25npathi/imdb-dataset-of-50k-movie-reviews`

[2]Link towards YELP dataset `https://www.kaggle.com/datasets/ilhamfp31/yelp-review-dataset`

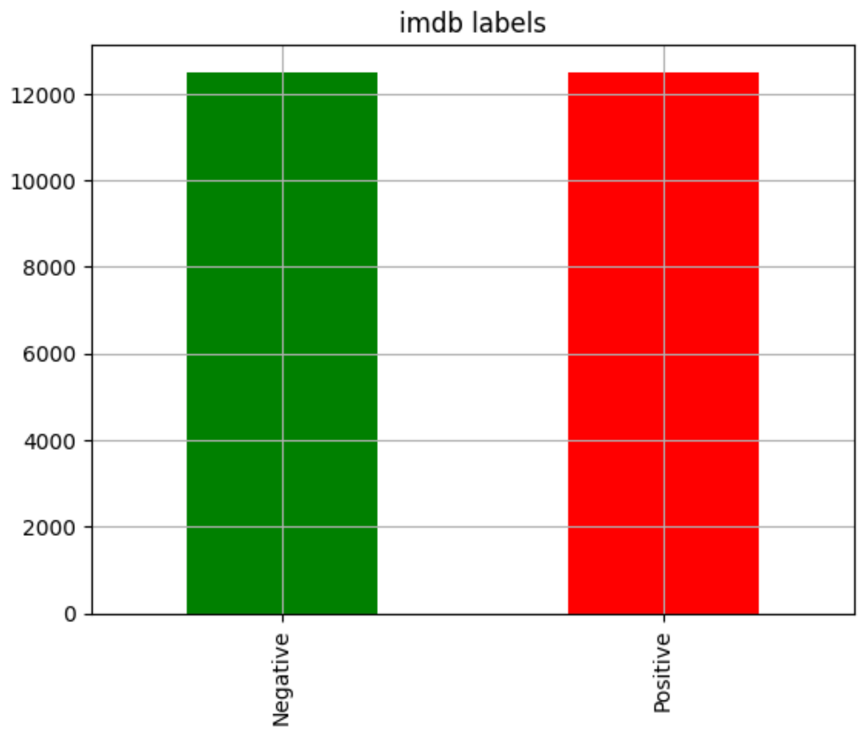


Figure 3.1: IMDB reviews data distribution

(c) Neutral

(d) Positive

(e) Extremely Positive

Data distribution is balanced, with 8101, 8067, 8064, 7899, and 7870 examples of extremely positive, positive, neutral, negative, and extremely negative classes, respectively, as shown in Figure 3.2.

3. Corona tweets

The corona tweets[3] dataset has 41157 tweets. It has five labels:

(a) Extremely Negative

[3]Link towards corona tweets dataset `https://www.kaggle.com/datasets/datatattle/covid-19-nlp-text-classification`

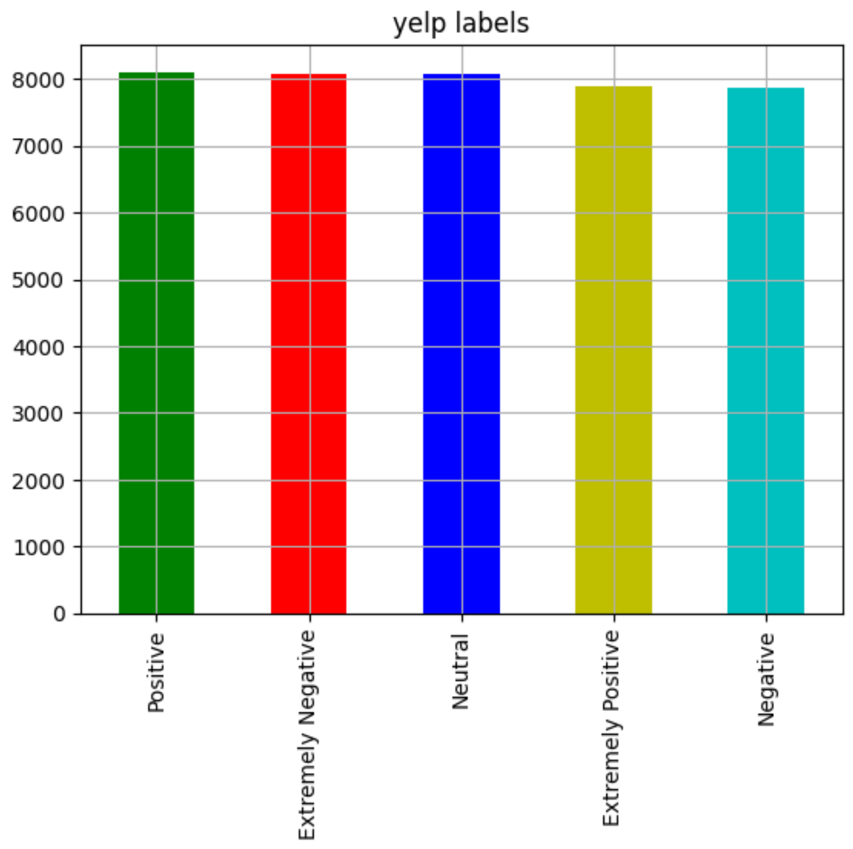


Figure 3.2: YELP reviews data distribution

(b) Negative

(c) Neutral

(d) Positive

(e) Extremely Positive

Data distribution is somewhat balanced, with 6624, 11422, 7713, 9917, and 7870 examples of extremely positive, positive, neutral, negative, and extremely negative classes, respectively, as shown in Figure 3.3.

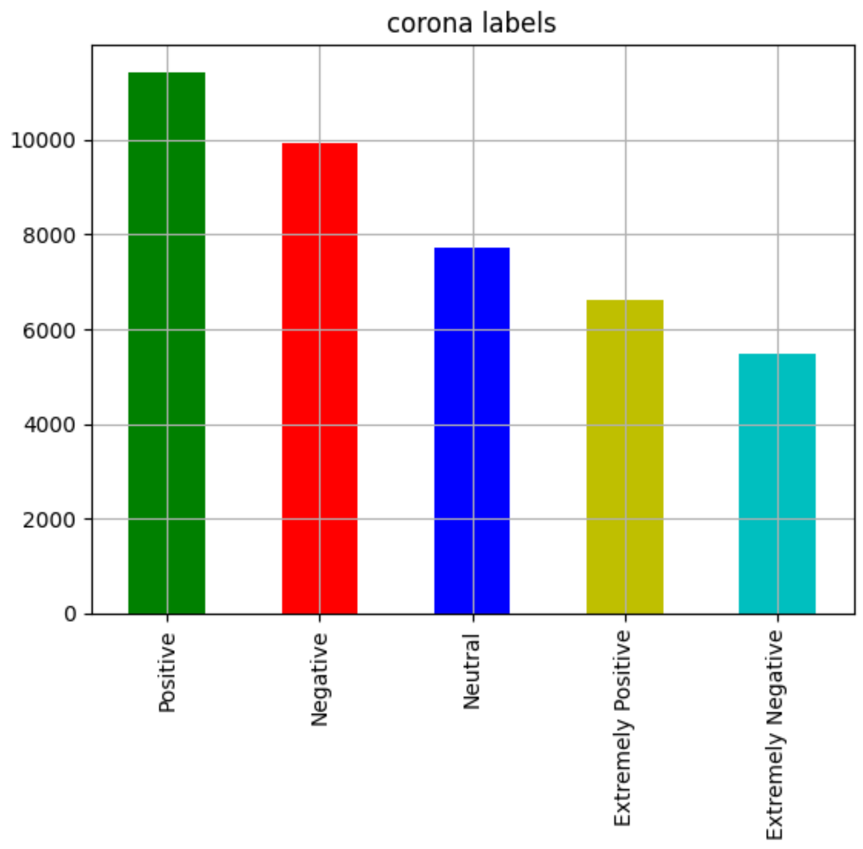
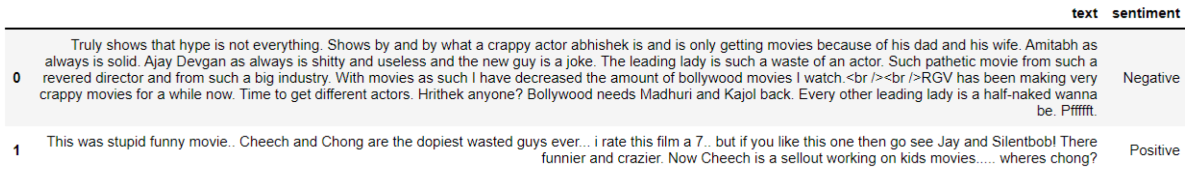


Figure 3.3: Corona tweets data distribution

3.1 Data preprocessing

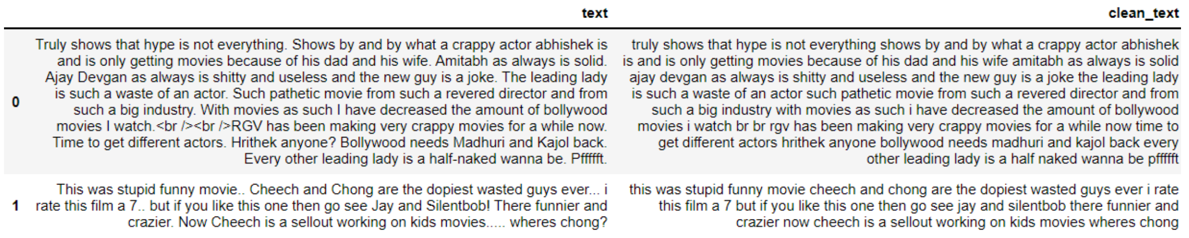
Before text is used for vectorization, preprocessing steps are applied to the raw text. For the IMDB reviews and YELP reviews, the same preprocessing techniques have been applied, in contrast to the corona tweets dataset, which had one more step because the scraped text on Twitter comes with hashtags. In the case of regular text, regex is used to clean all non-alphabetic and all non-numeric letters. All letters are lower-cased. For tweets, hashtags are removed using the tweet preprocessor library.
Two forms of text are saved: clean text and tokenized clean text, which used NLTK tokenizer to split the clean text into tokens (words). Either tokenized text is used as the input or the cleaned text, depending on the vectorization.

In Figure 3.4, Figure 3.5, and Figure 3.6 we have examples of raw, pre-processed, and tokenized dataset from the IMDB dataset, respectively.



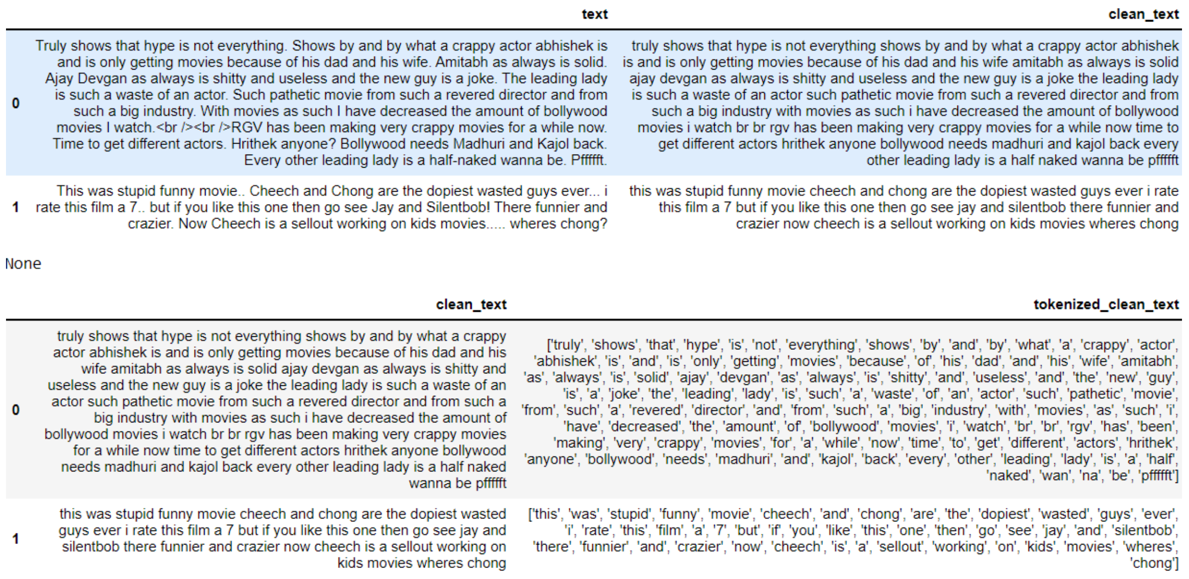
	text	sentiment
0	Truly shows that hype is not everything. Shows by and by what a crappy actor abhishek is and is only getting movies because of his dad and his wife. Amitabh as always is solid. Ajay Devgan as always is shitty and useless and the new guy is a joke. The leading lady is such a waste of an actor. Such pathetic movie from such a revered director and from such a big industry. With movies as such I have decreased the amount of bollywood movies I watch.<br /><br />RGV has been making very crappy movies for a while now. Time to get different actors. Hrithek anyone? Bollywood needs Madhuri and Kajol back. Every other leading lady is a half-naked wanna be. Pffffft.	Negative
1	This was stupid funny movie.. Cheech and Chong are the dopiest wasted guys ever... i rate this film a 7.. but if you like this one then go see Jay and Silentbob! There funnier and crazier. Now Cheech is a sellout working on kids movies..... wheres chong?	Positive

Figure 3.4: IMDB raw text



	text	clean_text
0	Truly shows that hype is not everything. Shows by and by what a crappy actor abhishek is and is only getting movies because of his dad and his wife. Amitabh as always is solid. Ajay Devgan as always is shitty and useless and the new guy is a joke. The leading lady is such a waste of an actor. Such pathetic movie from such a revered director and from such a big industry. With movies as such I have decreased the amount of bollywood movies I watch.<br /><br />RGV has been making very crappy movies for a while now. Time to get different actors. Hrithek anyone? Bollywood needs Madhuri and Kajol back. Every other leading lady is a half-naked wanna be. Pffffft.	truly shows that hype is not everything shows by and by what a crappy actor abhishek is and is only getting movies because of his dad and his wife amitabh as always is solid ajay devgan as always is shitty and useless and the new guy is a joke the leading lady is such a waste of an actor such pathetic movie from such a revered director and from such a big industry with movies as such i have decreased the amount of bollywood movies i watch br br rgv has been making very crappy movies for a while now time to get different actors hrithek anyone bollywood needs madhuri and kajol back every other leading lady is a half naked wanna be pffffft
1	This was stupid funny movie.. Cheech and Chong are the dopiest wasted guys ever... i rate this film a 7.. but if you like this one then go see Jay and Silentbob! There funnier and crazier. Now Cheech is a sellout working on kids movies..... wheres chong?	this was stupid funny movie cheech and chong are the dopiest wasted guys ever i rate this film a 7 but if you like this one then go see jay and silentbob there funnier and crazier now cheech is a sellout working on kids movies wheres chong

Figure 3.5: IMDB preprocessed text



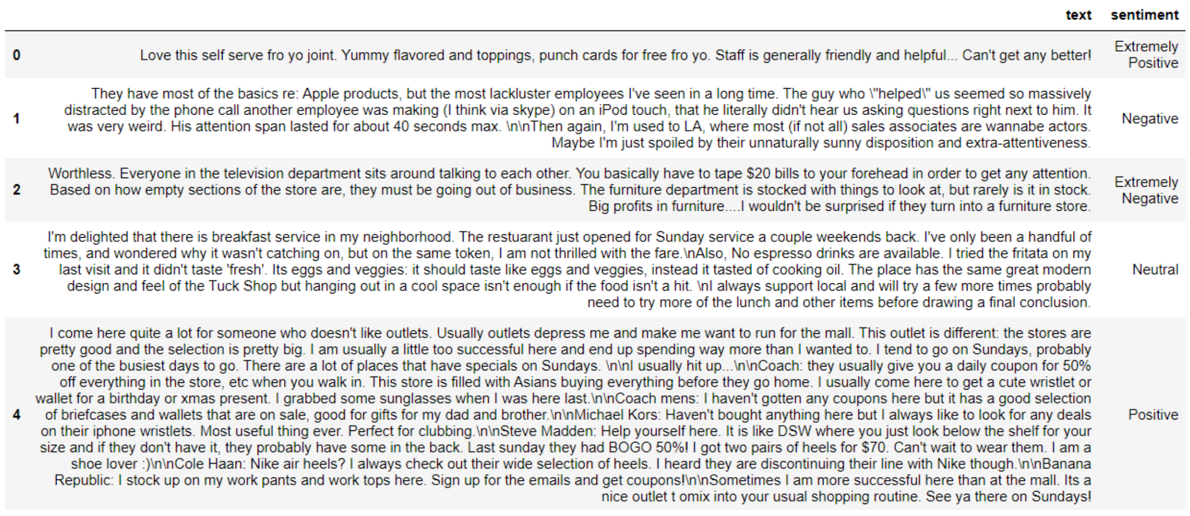
	text	clean_text
0	Truly shows that hype is not everything. Shows by and by what a crappy actor abhishek is and is only getting movies because of his dad and his wife. Amitabh as always is solid. Ajay Devgan as always is shitty and useless and the new guy is a joke. The leading lady is such a waste of an actor. Such pathetic movie from such a revered director and from such a big industry. With movies as such I have decreased the amount of bollywood movies I watch.<br /><br />RGV has been making very crappy movies for a while now. Time to get different actors. Hrithek anyone? Bollywood needs Madhuri and Kajol back. Every other leading lady is a half-naked wanna be. Pffffft.	truly shows that hype is not everything shows by and by what a crappy actor abhishek is and is only getting movies because of his dad and his wife amitabh as always is solid ajay devgan as always is shitty and useless and the new guy is a joke the leading lady is such a waste of an actor such pathetic movie from such a revered director and from such a big industry with movies as such i have decreased the amount of bollywood movies i watch br br rgv has been making very crappy movies for a while now time to get different actors hrithek anyone bollywood needs madhuri and kajol back every other leading lady is a half naked wanna be pffffft
1	This was stupid funny movie.. Cheech and Chong are the dopiest wasted guys ever... i rate this film a 7.. but if you like this one then go see Jay and Silentbob! There funnier and crazier. Now Cheech is a sellout working on kids movies..... wheres chong?	this was stupid funny movie cheech and chong are the dopiest wasted guys ever i rate this film a 7 but if you like this one then go see jay and silentbob there funnier and crazier now cheech is a sellout working on kids movies wheres chong

```
None
```

	clean_text	tokenized_clean_text
0	truly shows that hype is not everything shows by and by what a crappy actor abhishek is and is only getting movies because of his dad and his wife amitabh as always is solid ajay devgan as always is shitty and useless and the new guy is a joke the leading lady is such a waste of an actor such pathetic movie from such a revered director and from such a big industry with movies as such i have decreased the amount of bollywood movies i watch br br rgv has been making very crappy movies for a while now time to get different actors hrithek anyone bollywood needs madhuri and kajol back every other leading lady is a half naked wanna be pffffft	['truly', 'shows', 'that', 'hype', 'is', 'not', 'everything', 'shows', 'by', 'and', 'by', 'what', 'a', 'crappy', 'actor', 'abhishek', 'is', 'and', 'is', 'only', 'getting', 'movies', 'because', 'of', 'his', 'dad', 'and', 'his', 'wife', 'amitabh', 'as', 'always', 'is', 'solid', 'ajay', 'devgan', 'as', 'always', 'is', 'shitty', 'and', 'useless', 'and', 'the', 'new', 'guy', 'is', 'a', 'joke', 'the', 'leading', 'lady', 'is', 'such', 'a', 'waste', 'of', 'an', 'actor', 'such', 'pathetic', 'movie', 'from', 'such', 'a', 'revered', 'director', 'and', 'from', 'such', 'a', 'big', 'industry', 'with', 'movies', 'as', 'such', 'i', 'have', 'decreased', 'the', 'amount', 'of', 'bollywood', 'movies', 'i', 'watch', 'br', 'br', 'rgv', 'has', 'been', 'making', 'very', 'crappy', 'movies', 'for', 'a', 'while', 'now', 'time', 'to', 'get', 'different', 'actors', 'hrithek', 'anyone', 'bollywood', 'needs', 'madhuri', 'and', 'kajol', 'back', 'every', 'other', 'leading', 'lady', 'is', 'a', 'half', 'naked', 'wan', 'na', 'be', 'pffffft']
1	this was stupid funny movie cheech and chong are the dopiest wasted guys ever i rate this film a 7 but if you like this one then go see jay and silentbob there funnier and crazier now cheech is a sellout working on kids movies wheres chong	['this', 'was', 'stupid', 'funny', 'movie', 'cheech', 'and', 'chong', 'are', 'the', 'dopiest', 'wasted', 'guys', 'ever', 'i', 'rate', 'this', 'film', 'a', '7', 'but', 'if', 'you', 'like', 'this', 'one', 'then', 'go', 'see', 'jay', 'and', 'silentbob', 'there', 'funnier', 'and', 'crazier', 'now', 'cheech', 'is', 'a', 'sellout', 'working', 'on', 'kids', 'movies', 'wheres', 'chong']

Figure 3.6: IMDB tokenized text

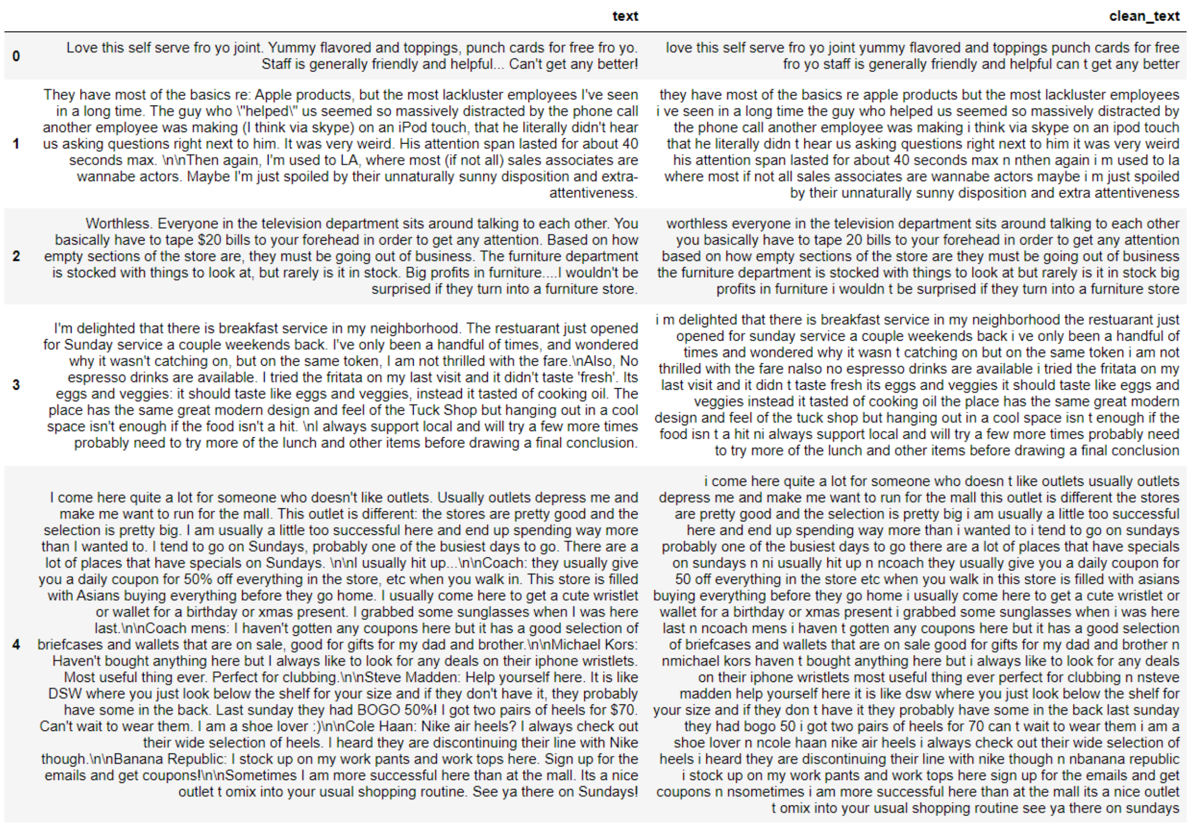
In Figure 3.7, Figure 3.8, and Figure 3.9 we have examples of raw, preprocessed, and tokenized dataset from the YELP dataset, respectively.



	text	sentiment
0	Love this self serve fro yo joint. Yummy flavored and toppings, punch cards for free fro yo. Staff is generally friendly and helpful... Can't get any better!	Extremely Positive
1	They have most of the basics re: Apple products, but the most lackluster employees I've seen in a long time. The guy who \"helped\" us seemed so massively distracted by the phone call another employee was making (I think via skype) on an iPod touch, that he literally didn't hear us asking questions right next to him. It was very weird. His attention span lasted for about 40 seconds max. \n\nThen again, I'm used to LA, where most (if not all) sales associates are wannabe actors. Maybe I'm just spoiled by their unnaturally sunny disposition and extra-attentiveness.	Negative
2	Worthless. Everyone in the television department sits around talking to each other. You basically have to tape $20 bills to your forehead in order to get any attention. Based on how empty sections of the store are, they must be going out of business. The furniture department is stocked with things to look at, but rarely is it in stock. Big profits in furniture....I wouldn't be surprised if they turn into a furniture store.	Extremely Negative
3	I'm delighted that there is breakfast service in my neighborhood. The restuarant just opened for Sunday service a couple weekends back. I've only been a handful of times, and wondered why it wasn't catching on, but on the same token, I am not thrilled with the fare.\nAlso, No espresso drinks are available. I tried the fritata on my last visit and it didn't taste 'fresh'. Its eggs and veggies: it should taste like eggs and veggies, instead it tasted of cooking oil. The place has the same great modern design and feel of the Tuck Shop but hanging out in a cool space isn't enough if the food isn't a hit. \nI always support local and will try a few more times probably need to try more of the lunch and other items before drawing a final conclusion.	Neutral
4	I come here quite a lot for someone who doesn't like outlets. Usually outlets depress me and make me want to run for the mall. This outlet is different: the stores are pretty good and the selection is pretty big. I am usually a little too successful here and end up spending way more than I wanted to. I tend to go on Sundays, probably one of the busiest days to go. There are a lot of places that have specials on Sundays. \n\nI usually hit up...\n\nCoach: they usually give you a daily coupon for 50% off everything in the store, etc when you walk in. This store is filled with Asians buying everything before they go home. I usually come here to get a cute wristlet or wallet for a birthday or xmas present. I grabbed some sunglasses when I was here last.\n\nCoach mens: I haven't gotten any coupons here but it has a good selection of briefcases and wallets that are on sale, good for gifts for my dad and brother.\n\nMichael Kors: Haven't bought anything here but I always like to look for any deals on their iphone wristlets. Most useful thing ever. Perfect for clubbing.\n\nSteve Madden: Help yourself here. It is like DSW where you just look below the shelf for your size and if they don't have it, they probably have some in the back. Last sunday they had BOGO 50%! I got two pairs of heels for $70. Can't wait to wear them. I am a shoe lover :)\n\nCole Haan: Nike air heels? I always check out their wide selection of heels. I heard they are discontinuing their line with Nike though.\n\nBanana Republic: I stock up on my work pants and work tops here. Sign up for the emails and get coupons!\n\nSometimes I am more successful here than at the mall. Its a nice outlet t omix into your usual shopping routine. See ya there on Sundays!	Positive

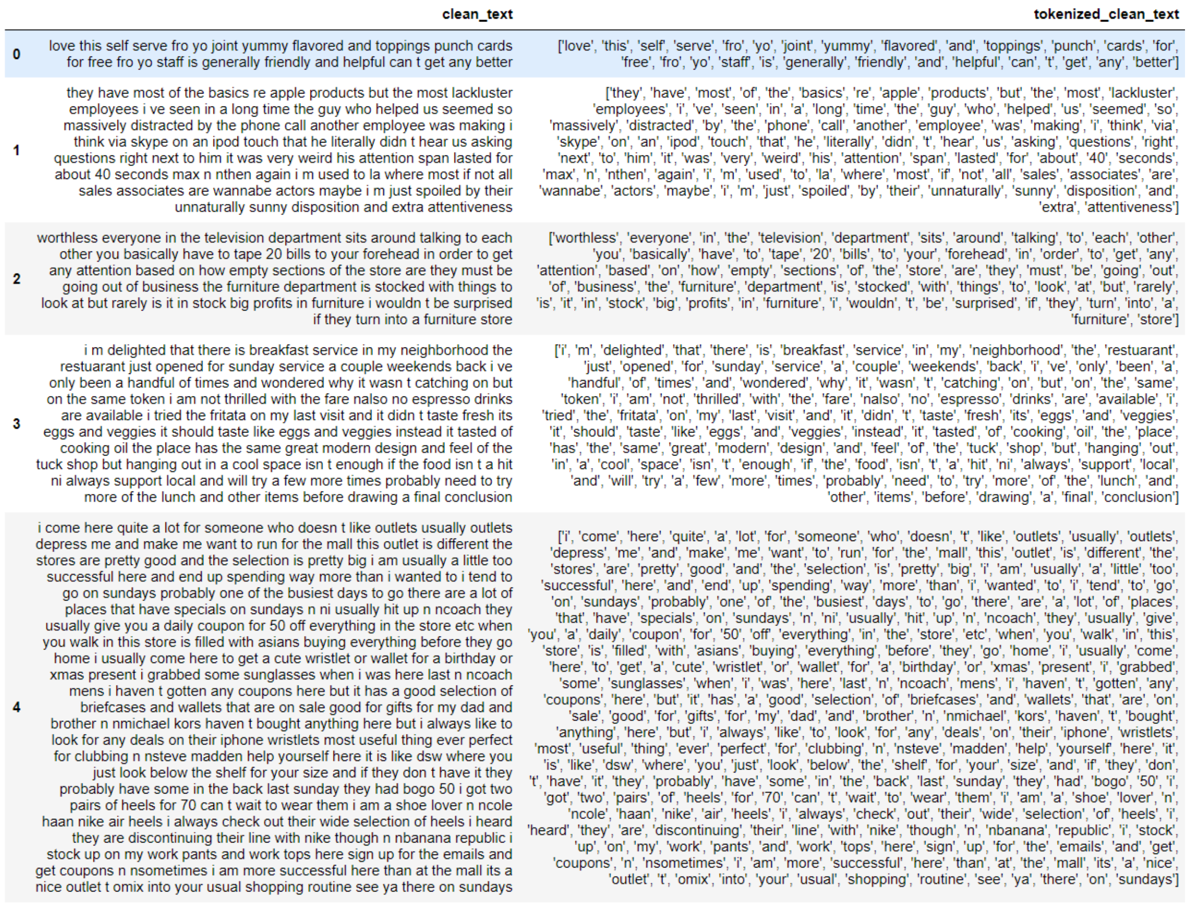
Figure 3.7: YELP raw text

In Figure 3.10, Figure 3.11, and Figure 3.12 we have examples of raw, preprocessed, and tokenized dataset from the Corona tweets dataset, respectively.



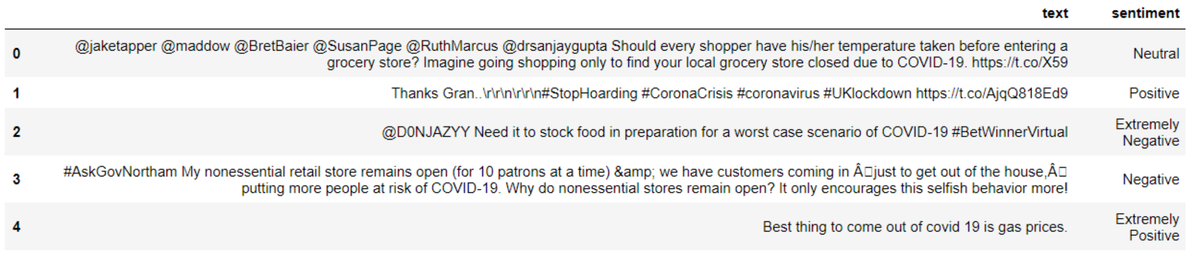
	text	clean_text
0	Love this self serve fro yo joint. Yummy flavored and toppings, punch cards for free fro yo. Staff is generally friendly and helpful... Can't get any better!	love this self serve fro yo joint yummy flavored and toppings punch cards for free fro yo staff is generally friendly and helpful can t get any better
1	They have most of the basics re: Apple products, but the most lackluster employees I've seen in a long time. The guy who \"helped\" us seemed so massively distracted by the phone call another employee was making (I think via skype) on an iPod touch, that he literally didn't hear us asking questions right next to him. It was very weird. His attention span lasted for about 40 seconds max. \n\nThen again, I'm used to LA, where most (if not all) sales associates are wannabe actors. Maybe I'm just spoiled by their unnaturally sunny disposition and extra-attentiveness.	they have most of the basics re apple products but the most lackluster employees i ve seen in a long time the guy who helped us seemed so massively distracted by the phone call another employee was making i think via skype on an ipod touch that he literally didn t hear us asking questions right next to him it was very weird his attention span lasted for about 40 seconds max n nthen again i m used to la where most if not all sales associates are wannabe actors maybe i m just spoiled by their unnaturally sunny disposition and extra attentiveness
2	Worthless. Everyone in the television department sits around talking to each other. You basically have to tape $20 bills to your forehead in order to get any attention. Based on how empty sections of the store are, they must be going out of business. The furniture department is stocked with things to look at, but rarely is it in stock. Big profits in furniture....I wouldn't be surprised if they turn into a furniture store.	worthless everyone in the television department sits around talking to each other you basically have to tape 20 bills to your forehead in order to get any attention based on how empty sections of the store are they must be going out of business the furniture department is stocked with things to look at but rarely is it in stock big profits in furniture i wouldn t be surprised if they turn into a furniture store
3	I'm delighted that there is breakfast service in my neighborhood. The restuarant just opened for Sunday service a couple weekends back. I've only been a handful of times, and wondered why it wasn't catching on, but on the same token, I am not thrilled with the fare.\nAlso, No espresso drinks are available. I tried the fritata on my last visit and it didn't taste 'fresh'. Its eggs and veggies: it should taste like eggs and veggies, instead it tasted of cooking oil. The place has the same great modern design and feel of the Tuck Shop but hanging out in a cool space isn't enough if the food isn't a hit. \nI always support local and will try a few more times probably need to try more of the lunch and other items before drawing a final conclusion.	i m delighted that there is breakfast service in my neighborhood the restuarant just opened for sunday service a couple weekends back i ve only been a handful of times and wondered why it wasn t catching on but on the same token i am not thrilled with the fare nalso no espresso drinks are available i tried the fritata on my last visit and it didn t taste fresh its eggs and veggies it should taste like eggs and veggies instead it tasted of cooking oil the place has the same great modern design and feel of the tuck shop but hanging out in a cool space isn t enough if the food isn t a hit ni always support local and will try a few more times probably need to try more of the lunch and other items before drawing a final conclusion
4	I come here quite a lot for someone who doesn't like outlets. Usually outlets depress me and make me want to run for the mall. This outlet is different: the stores are pretty good and the selection is pretty big. I am usually a little too successful here and end up spending way more than I wanted to. I tend to go on Sundays, probably one of the busiest days to go. There are a lot of places that have specials on Sundays. \n\nI usually hit up...\n\nCoach: they usually give you a daily coupon for 50% off everything in the store, etc when you walk in. This store is filled with Asians buying everything before they go home. I usually come here to get a cute wristlet or wallet for a birthday or xmas present. I grabbed some sunglasses when I was here last.\n\nCoach mens: I haven't gotten any coupons here but it has a good selection of briefcases and wallets that are on sale, good for gifts for my dad and brother.\n\nMichael Kors: Haven't bought anything here but I always like to look for any deals on their iphone wristlets. Most useful thing ever. Perfect for clubbing.\n\nSteve Madden: Help yourself here. It is like DSW where you just look below the shelf for your size and if they don't have it, they probably have some in the back. Last sunday they had BOGO 50%! I got two pairs of heels for $70. Can't wait to wear them. I am a shoe lover :)\n\nCole Haan: Nike air heels? I always check out their wide selection of heels. I heard they are discontinuing their line with Nike though.\n\nBanana Republic: I stock up on my work pants and work tops here. Sign up for the emails and get coupons!\n\nSometimes I am more successful here than at the mall. Its a nice outlet t omix into your usual shopping routine. See ya there on Sundays!	i come here quite a lot for someone who doesn t like outlets usually outlets depress me and make me want to run for the mall this outlet is different the stores are pretty good and the selection is pretty big i am usually a little too successful here and end up spending way more than i wanted to i tend to go on sundays probably one of the busiest days to go there are a lot of places that have specials on sundays n ni usually hit up n ncoach they usually give you a daily coupon for 50 off everything in the store etc when you walk in this store is filled with asians buying everything before they go home i usually come here to get a cute wristlet or wallet for a birthday or xmas present i grabbed some sunglasses when i was here last n ncoach mens i haven t gotten any coupons here but it has a good selection of briefcases and wallets that are on sale good for gifts for my dad and brother n nmichael kors haven t bought anything here but i always like to look for any deals on their iphone wristlets most useful thing ever perfect for clubbing n nsteve madden help yourself here it is like dsw where you just look below the shelf for your size and if they don t have it they probably have some in the back last sunday they had bogo 50 i got two pairs of heels for 70 can t wait to wear them i am a shoe lover n ncole haan nike air heels i always check out their wide selection of heels i heard they are discontinuing their line with nike though n nbanana republic i stock up on my work pants and work tops here sign up for the emails and get coupons n nsometimes i am more successful here than at the mall its a nice outlet t omix into your usual shopping routine see ya there on sundays

Figure 3.8: YELP preprocessed text



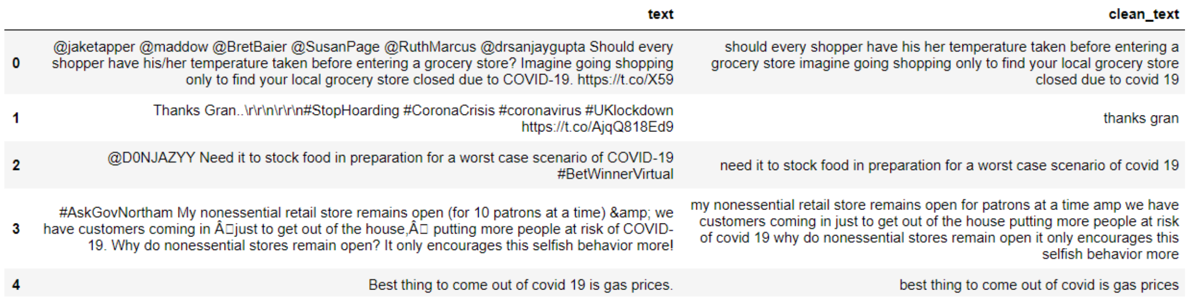
	clean_text	tokenized_clean_text
0	love this self serve fro yo joint yummy flavored and toppings punch cards for free fro yo staff is generally friendly and helpful can t get any better	['love', 'this', 'self', 'serve', 'fro', 'yo', 'joint', 'yummy', 'flavored', 'and', 'toppings', 'punch', 'cards', 'for', 'free', 'fro', 'yo', 'staff', 'is', 'generally', 'friendly', 'and', 'helpful', 'can', 't', 'get', 'any', 'better']
1	they have most of the basics re apple products but the most lackluster employees i ve seen in a long time the guy who helped us seemed so massively distracted by the phone call another employee was making i think via skype on an ipod touch that he literally didn t hear us asking questions right next to him it was very weird his attention span lasted for about 40 seconds max n nthen again i m used to la where most if not all sales associates are wannabe actors maybe i m just spoiled by their unnaturally sunny disposition and extra attentiveness	['they', 'have', 'most', 'of', 'the', 'basics', 're', 'apple', 'products', 'but', 'the', 'most', 'lackluster', 'employees', 'i', 've', 'seen', 'in', 'a', 'long', 'time', 'the', 'guy', 'who', 'helped', 'us', 'seemed', 'so', 'massively', 'distracted', 'by', 'the', 'phone', 'call', 'another', 'employee', 'was', 'making', 'i', 'think', 'via', 'skype', 'on', 'an', 'ipod', 'touch', 'that', 'he', 'literally', 'didn', 't', 'hear', 'us', 'asking', 'questions', 'right', 'next', 'to', 'him', 'it', 'was', 'very', 'weird', 'his', 'attention', 'span', 'lasted', 'for', 'about', '40', 'seconds', 'max', 'n', 'nthen', 'again', 'i', 'm', 'used', 'to', 'la', 'where', 'most', 'if', 'not', 'all', 'sales', 'associates', 'are', 'wannabe', 'actors', 'maybe', 'i', 'm', 'just', 'spoiled', 'by', 'their', 'unnaturally', 'sunny', 'disposition', 'and', 'extra', 'attentiveness']
2	worthless everyone in the television department sits around talking to each other you basically have to tape 20 bills to your forehead in order to get any attention based on how empty sections of the store are they must be going out of business the furniture department is stocked with things to look at but rarely is it in stock big profits in furniture i wouldn t be surprised if they turn into a furniture store	['worthless', 'everyone', 'in', 'the', 'television', 'department', 'sits', 'around', 'talking', 'to', 'each', 'other', 'you', 'basically', 'have', 'to', 'tape', '20', 'bills', 'to', 'your', 'forehead', 'in', 'order', 'to', 'get', 'any', 'attention', 'based', 'on', 'how', 'empty', 'sections', 'of', 'the', 'store', 'are', 'they', 'must', 'be', 'going', 'out', 'of', 'business', 'the', 'furniture', 'department', 'is', 'stocked', 'with', 'things', 'to', 'look', 'at', 'but', 'rarely', 'is', 'it', 'in', 'stock', 'big', 'profits', 'in', 'furniture', 'i', 'wouldn', 't', 'be', 'surprised', 'if', 'they', 'turn', 'into', 'a', 'furniture', 'store']
3	i m delighted that there is breakfast service in my neighborhood the restuarant just opened for sunday service a couple weekends back i ve only been a handful of times and wondered why it wasn t catching on but on the same token i am not thrilled with the fare nalso no espresso drinks are available i tried the fritata on my last visit and it didn t taste fresh its eggs and veggies it should taste like eggs and veggies instead it tasted of cooking oil the place has the same great modern design and feel of the tuck shop but hanging out in a cool space isn t enough if the food isn t a hit ni always support local and will try a few more times probably need to try more of the lunch and other items before drawing a final conclusion	['i', 'm', 'delighted', 'that', 'there', 'is', 'breakfast', 'service', 'in', 'my', 'neighborhood', 'the', 'restuarant', 'just', 'opened', 'for', 'sunday', 'service', 'a', 'couple', 'weekends', 'back', 'i', 've', 'only', 'been', 'a', 'handful', 'of', 'times', 'and', 'wondered', 'why', 'it', 'wasn', 't', 'catching', 'on', 'but', 'on', 'the', 'same', 'token', 'i', 'am', 'not', 'thrilled', 'with', 'the', 'fare', 'nalso', 'no', 'espresso', 'drinks', 'are', 'available', 'i', 'tried', 'the', 'fritata', 'on', 'my', 'last', 'visit', 'and', 'it', 'didn', 't', 'taste', 'fresh', 'its', 'eggs', 'and', 'veggies', 'it', 'should', 'taste', 'like', 'eggs', 'and', 'veggies', 'instead', 'it', 'tasted', 'of', 'cooking', 'oil', 'the', 'place', 'has', 'the', 'same', 'great', 'modern', 'design', 'and', 'feel', 'of', 'the', 'tuck', 'shop', 'but', 'hanging', 'out', 'in', 'a', 'cool', 'space', 'isn', 't', 'enough', 'if', 'the', 'food', 'isn', 't', 'a', 'hit', 'ni', 'always', 'support', 'local', 'and', 'will', 'try', 'a', 'few', 'more', 'times', 'probably', 'need', 'to', 'try', 'more', 'of', 'the', 'lunch', 'and', 'other', 'items', 'before', 'drawing', 'a', 'final', 'conclusion']
4	i come here quite a lot for someone who doesn t like outlets usually outlets depress me and make me want to run for the mall this outlet is different the stores are pretty good and the selection is pretty big i am usually a little too successful here and end up spending way more than i wanted to i tend to go on sundays probably one of the busiest days to go there are a lot of places that have specials on sundays n ni usually hit up n ncoach they usually give you a daily coupon for 50 off everything in the store etc when you walk in this store is filled with asians buying everything before they go home i usually come here to get a cute wristlet or wallet for a birthday or xmas present i grabbed some sunglasses when i was here last n ncoach mens i haven t gotten any coupons here but it has a good selection of briefcases and wallets that are on sale good for gifts for my dad and brother n nmichael kors haven t bought anything here but i always like to look for any deals on their iphone wristlets most useful thing ever perfect for clubbing n nsteve madden help yourself here it is like dsw where you just look below the shelf for your size and if they don t have it they probably have some in the back last sunday they had bogo 50 i got two pairs of heels for 70 can t wait to wear them i am a shoe lover n ncole haan nike air heels i always check out their wide selection of heels i heard they are discontinuing their line with nike though n nbanana republic i stock up on my work pants and work tops here sign up for the emails and get coupons n nsometimes i am more successful here than at the mall its a nice outlet t omix into your usual shopping routine see ya there on sundays	['i', 'come', 'here', 'quite', 'a', 'lot', 'for', 'someone', 'who', 'doesn', 't', 'like', 'outlets', 'usually', 'outlets', 'depress', 'me', 'and', 'make', 'me', 'want', 'to', 'run', 'for', 'the', 'mall', 'this', 'outlet', 'is', 'different', 'the', 'stores', 'are', 'pretty', 'good', 'and', 'the', 'selection', 'is', 'pretty', 'big', 'i', 'am', 'usually', 'a', 'little', 'too', 'successful', 'here', 'and', 'end', 'up', 'spending', 'way', 'more', 'than', 'i', 'wanted', 'to', 'i', 'tend', 'to', 'go', 'on', 'sundays', 'probably', 'one', 'of', 'the', 'busiest', 'days', 'to', 'go', 'there', 'are', 'a', 'lot', 'of', 'places', 'that', 'have', 'specials', 'on', 'sundays', 'n', 'ni', 'usually', 'hit', 'up', 'n', 'ncoach', 'they', 'usually', 'give', 'you', 'a', 'daily', 'coupon', 'for', '50', 'off', 'everything', 'in', 'the', 'store', 'etc', 'when', 'you', 'walk', 'in', 'this', 'store', 'is', 'filled', 'with', 'asians', 'buying', 'everything', 'before', 'they', 'go', 'home', 'i', 'usually', 'come', 'here', 'to', 'get', 'a', 'cute', 'wristlet', 'or', 'wallet', 'for', 'a', 'birthday', 'or', 'xmas', 'present', 'i', 'grabbed', 'some', 'sunglasses', 'when', 'i', 'was', 'here', 'last', 'n', 'ncoach', 'mens', 'i', 'haven', 't', 'gotten', 'any', 'coupons', 'here', 'but', 'it', 'has', 'a', 'good', 'selection', 'of', 'briefcases', 'and', 'wallets', 'that', 'are', 'on', 'sale', 'good', 'for', 'gifts', 'for', 'my', 'dad', 'and', 'brother', 'n', 'nmichael', 'kors', 'haven', 't', 'bought', 'anything', 'here', 'but', 'i', 'always', 'like', 'to', 'look', 'for', 'any', 'deals', 'on', 'their', 'iphone', 'wristlets', 'most', 'useful', 'thing', 'ever', 'perfect', 'for', 'clubbing', 'n', 'nsteve', 'madden', 'help', 'yourself', 'here', 'it', 'is', 'like', 'dsw', 'where', 'you', 'just', 'look', 'below', 'the', 'shelf', 'for', 'your', 'size', 'and', 'if', 'they', 'don', 't', 'have', 'it', 'they', 'probably', 'have', 'some', 'in', 'the', 'back', 'last', 'sunday', 'they', 'had', 'bogo', '50', 'i', 'got', 'two', 'pairs', 'of', 'heels', 'for', '70', 'can', 't', 'wait', 'to', 'wear', 'them', 'i', 'am', 'a', 'shoe', 'lover', 'n', 'ncole', 'haan', 'nike', 'air', 'heels', 'i', 'always', 'check', 'out', 'their', 'wide', 'selection', 'of', 'heels', 'i', 'heard', 'they', 'are', 'discontinuing', 'their', 'line', 'with', 'nike', 'though', 'n', 'nbanana', 'republic', 'i', 'stock', 'up', 'on', 'my', 'work', 'pants', 'and', 'work', 'tops', 'here', 'sign', 'up', 'for', 'the', 'emails', 'and', 'get', 'coupons', 'n', 'nsometimes', 'i', 'am', 'more', 'successful', 'here', 'than', 'at', 'the', 'mall', 'its', 'a', 'nice', 'outlet', 't', 'omix', 'into', 'your', 'usual', 'shopping', 'routine', 'see', 'ya', 'there', 'on', 'sundays']

Figure 3.9: YELP tokenized text



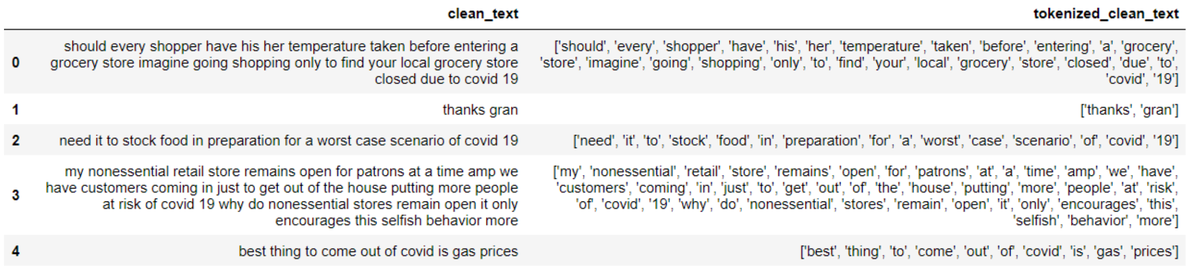
	text	sentiment
0	@jaketapper @maddow @BretBaier @SusanPage @RuthMarcus @drsanjaygupta Should every shopper have his/her temperature taken before entering a grocery store? Imagine going shopping only to find your local grocery store closed due to COVID-19. https://t.co/X59	Neutral
1	Thanks Gran..\r\r\n\r\r\n#StopHoarding #CoronaCrisis #coronavirus #UKlockdown https://t.co/AjqQ818Ed9	Positive
2	@D0NJAZYY Need it to stock food in preparation for a worst case scenario of COVID-19 #BetWinnerVirtual	Extremely Negative
3	#AskGovNortham My nonessential retail store remains open (for 10 patrons at a time) &amp; we have customers coming in Â□just to get out of the house,Â□ putting more people at risk of COVID-19. Why do nonessential stores remain open? It only encourages this selfish behavior more!	Negative
4	Best thing to come out of covid 19 is gas prices.	Extremely Positive

Figure 3.10: Corona tweets raw text



	text	clean_text
0	@jaketapper @maddow @BretBaier @SusanPage @RuthMarcus @drsanjaygupta Should every shopper have his/her temperature taken before entering a grocery store? Imagine going shopping only to find your local grocery store closed due to COVID-19. https://t.co/X59	should every shopper have his her temperature taken before entering a grocery store imagine going shopping only to find your local grocery store closed due to covid 19
1	Thanks Gran..\r\r\n\r\r\n#StopHoarding #CoronaCrisis #coronavirus #UKlockdown https://t.co/AjqQ818Ed9	thanks gran
2	@D0NJAZYY Need it to stock food in preparation for a worst case scenario of COVID-19 #BetWinnerVirtual	need it to stock food in preparation for a worst case scenario of covid 19
3	#AskGovNortham My nonessential retail store remains open (for 10 patrons at a time) &amp; we have customers coming in Â□just to get out of the house,Â□ putting more people at risk of COVID-19. Why do nonessential stores remain open? It only encourages this selfish behavior more!	my nonessential retail store remains open for patrons at a time amp we have customers coming in just to get out of the house putting more people at risk of covid 19 why do nonessential stores remain open it only encourages this selfish behavior more
4	Best thing to come out of covid 19 is gas prices.	best thing to come out of covid is gas prices

Figure 3.11: Corona tweets preprocessed text



	clean_text	tokenized_clean_text
0	should every shopper have his her temperature taken before entering a grocery store imagine going shopping only to find your local grocery store closed due to covid 19	['should', 'every', 'shopper', 'have', 'his', 'her', 'temperature', 'taken', 'before', 'entering', 'a', 'grocery', 'store', 'imagine', 'going', 'shopping', 'only', 'to', 'find', 'your', 'local', 'grocery', 'store', 'closed', 'due', 'to', 'covid', '19']
1	thanks gran	['thanks', 'gran']
2	need it to stock food in preparation for a worst case scenario of covid 19	['need', 'it', 'to', 'stock', 'food', 'in', 'preparation', 'for', 'a', 'worst', 'case', 'scenario', 'of', 'covid', '19']
3	my nonessential retail store remains open for patrons at a time amp we have customers coming in just to get out of the house putting more people at risk of covid 19 why do nonessential stores remain open it only encourages this selfish behavior more	['my', 'nonessential', 'retail', 'store', 'remains', 'open', 'for', 'patrons', 'at', 'a', 'time', 'amp', 'we', 'have', 'customers', 'coming', 'in', 'just', 'to', 'get', 'out', 'of', 'the', 'house', 'putting', 'more', 'people', 'at', 'risk', 'of', 'covid', '19', 'why', 'do', 'nonessential', 'stores', 'remain', 'open', 'it', 'only', 'encourages', 'this', 'selfish', 'behavior', 'more']
4	best thing to come out of covid is gas prices	['best', 'thing', 'to', 'come', 'out', 'of', 'covid', 'is', 'gas', 'prices']

Figure 3.12: Corona tweets tokenized text

Chapter 4

Features

In order to use text for sentiment analysis (or for any task using machine learning), the text needs to be converted to vectors before passing it to the classification models. This process is known as vectorization. There are a couple of different vectorization techniques.

4.1 Bag of words - BOW

A bag of words [26] is a technique used in text modeling to extract features from the text. The idea here is to treat the words as a bag of words, with no notion of order and context.
Bag of words maps unique integer IDs between 1 and $|V|$, where V is vocabulary and $|V|$ is vocabulary size. Each document in a corpus c is converted into a vector of $|V|$ dimensions, where the i-th index is the number of times word w_i appear in a document.

Some advantages of BoW:

1. BoW is easy to implement and understand.
2. Documents having the same words will be closer than documents with different words.
3. Fixed length encoding.

Some disadvantages of BoW:

1. The size of the vocabulary increases the vector size, so sparsity becomes an issue.

2. It does not capture the similarity between different words. Sentences "I ran", "I run", and "I ate" will have the same distance between each pair of them.

3. It is hard to handle out-of-vocabulary words.

4. Word order information is lost.

4.2 Term Frequency - Inverse Document Frequency (TF-IDF)

Term frequency-inverse document frequency [27] consists of two parts:

1. Term Frequency (TF)

2. Inverse Document Frequency (IDF)

Term Frequency (TF)

Term frequency is a measure to count how many times a word appears in a document. In different document lengths, words may appear a different number of times in a document, based on the length, thus not giving a similar impact on a word appearing in a short versus a long document. That is why the score is normalized with the document length. For a given document d, and word t, the TF score is:

$$TF(t,d) = \frac{\text{Number of occurrences of the term t in document d}}{\text{Total number of words in document d}} \tag{4.1}$$

Inverse Document Frequency (IDF)

Inverse document frequency is a measure of how relevant a word is in a document. In terms of frequency, all words are equally important. For example, the most common words in English are ‘a‘, ‘the‘, etc., but they do not have any relevance (known as stop words). Taking that into consideration, IDF gives a low score to words that are very common in the collection of documents (corpus) and a high to words that are less common in a corpus.

$$IDF(t) = log(\frac{\text{Total number of documents containing word t}}{\text{Number of documents in corpus}}) \quad (4.2)$$

If a corpus c contains d documents and t number of different words, TF-IDF is a matrix of shape (d, t), where the i-th row represents the vectorization of the i-th document in a corpus, and the j-th column is the TF-IDF score of the j-th term across the collection.

Term Frequency - Inverse Document Frequency (TF-IDF)

Term frequency-inverse document frequency is a product of the previous two scores. TF-IDF score of a word t, in document d from corpus, is:

$$TFIDF(t, d) = TF(t, d) * IDF(d) \quad (4.3)$$

To illustrate the previous concepts we use the following corpus of documents:

1. Dog bites man.
2. Man bites dog.
3. Dog eats meat.
4. Man eats food.

TF-IDF scores for the previous corpus of documents are shown in Table 4.1.

Word	TF score	IDF score	TF-IDF score
dog	$1/3 = 0.33$	$log_2(4/3) = 0.4114$	$0.4114 * 0.33 = 0.136$
bites	$1/6 = 0.17$	$log_2(4/2) = 1$	$1 * 0.17 = 0.17$
man	0.33	$log_2(4/3) = 0.4114$	$0.4114 * 0.33 = 0.136$
eats	0.17	$log_2(4/2) = 1$	$1 * 0.17 = 0.17$
meat	$1/12 = 0.083$	$log_2(4/1) = 2$	$2 * 0.0083 = 0.17$
food	0.083	$log_2(4/1) = 2$	$2 * 0.0083 = 0.17$

Table 4.1: Example of TF-IDF values

TF-IDF score of the document ‘Dog bites man.’ from corpus, is shown in Table 4.2.

Dog	bites	man	eats	meat	food
0.136	0.17	0.136	0	0	0

Table 4.2: Example of TF-IDF vector

TF-IDF [26] is a naive approach to vectorization since it does not take into account the order of words. Two different sentences with the same words have the same vector representations. This is known as a Bag of Words (BoW). The drawback of TF-IDF:

- They can’t capture the relationship between words.
- The feature vectors are sparse and highly dimensional. Dimensionality increases with the size of the vocabulary, and most values are zero for any vector. This makes computation inefficient.
- They can’t handle out-of-vocabulary words.

4.3 Pretrained Word Embeddings

TF-IDF approach treated words as atomic units - there is no notion of similarity between words, as they are represented as indices in a vocabulary. This is a simple and robust approach, but very limited. The dimensionality of embedded space is high and sparse. Instead of using the TF-IDF approach, we can try to vectorize words more carefully to have low, dense embedding space with the notion of similarity. Some of these approaches are:

- Word2Vec
- GloVe
- FastText

4.3.1 Word2Vec

Word2Vec paper has been published in 2013 by Google researchers [21] and it was a state-of-the-art vectorization, opening a new chapter in word vectorization.

Model architecture

Base architecture for Word2Vec has been inspired by the feedforward neural net language model (NNLM) from [28]. The neural network has a couple of layers: input, projection, hidden, and output.

For the input, previous N words are encoded using 1-of-V coding, where V is the size of the vocabulary. The input layer is then projected to a projection layer P that has dimensionality NxD, using a shared projection matrix. Since only N words are taken, this is a cheap operation.

Two new architectures are proposed for learning distributed representation of words.

Continuous Bag-of-Words Model (CBOW)

CBOW [29] architecture, as shown in Figure 4.1, is similar to feedforward NNLM, except that the non-linear hidden layer is removed and the projection layer is shared for all words. It's called the bag-of-words model because the order of words does not influence the projection. In other terms, the context of a word is taken and based on context, the model tries to detect which word is it from the vocabulary. We use the following notation in the rest of this section:

- w_i: Word i from vocabulary V
- $V \in \mathbb{R}^{\mathrm{nx}|\mathrm{V}|}$: Input word matrix
- v_i: i-th column of V, the input vector representation of word w_i
- $U \in \mathbb{R}^{|\mathrm{V}|\mathrm{xn}}$: Output word matrix
- u_i: i-th row of U, the output vector representation of word w_i

We create two matrices $V \in \mathbb{R}^{\mathrm{nx}|\mathrm{V}|}$ and $U \in \mathbb{R}^{|\mathrm{V}|\mathrm{xn}}$, where n is an arbitrary size that defines the size of our embedding space. V is the input word matrix such that the i-th column of V is the n-dimensional embedded vector for word w_i when it is an input to this model. We denote this $n \times 1$ vector as v_i. Similarly, U is the output word matrix. The j-th row of U is an n-dimensional embedded vector for word w_j when it is an output of the model. We denote this row of U as u_j. Note that we do in fact learn two vectors for every word w_i (i.e. input word vector v_i and output word vector u_i).

Firstly, input words are converted to one hot vector of size m:

$$(x_{c-m}, ..., x_{c-1}, x_{c+1}, ..., x_{c+m}) \tag{4.4}$$

After getting one hot vectors, they are passed to the input word matrix V to get context

$$v_{c-m} = Vx_{c-m}, v_{c-m+1} = Vx_{c-m+1}, ..., v_{c+m} = Vx_{c+m} \tag{4.5}$$

All these context words are averaged to get one vector $\hat{v}$, where:

$$\hat{v} = \frac{v_{c-m} + v_{c-m+1} + ...v_{c+m-1} + v_{c+m}}{2m} \tag{4.6}$$

The output vector score is generated by passing averaged vector to the output matrix, $z = U\hat{v}$. The output vector is then passed through the softmax function to get the output probabilities, $\hat{y} = \text{softmax}(z)$. The output probabilities should represent the actual one-hot vector of the input.

For training this model, the cross-entropy loss function is used $H(\hat{y}, y)$. $H(\hat{y}, y) = -\sum_{j=1}^{||V|} y_j \log(\hat{y_j})$. Because y is one hot encoded, previous formula becomes: $H(\hat{y}, y) = -y_i \log(\hat{y_i})$. Index c is where one hot vector has 1. If $\hat{y_c} = 1$, perfect prediction, then $H(\hat{y}, y) = -1 \log(1) = 0$. $\hat{y_c} = 0.001$, very bad prediction, $H(\hat{y}, y) = -1 \log(0.001) \approx 4.605$. Thus cross entropy provides a good measure of distance. Now, the objective which is being optimized is

$$\begin{aligned} \textit{minimize } \mathrm{L} &= -\log P(w_c|w_{c-m}, ..., w_{c-1}, w_{c+1}, ..., w_{c+m}) \\ &= -\log P(u_c|\hat{v}) \\ &= -\log \frac{exp(u_c^T)}{\sum_{j=1}^{|V|} exp(u_j^T \hat{v})} \end{aligned} \tag{4.7}$$

The stochastic gradient is used to update all relevant word vectors u_c and v_j [29].

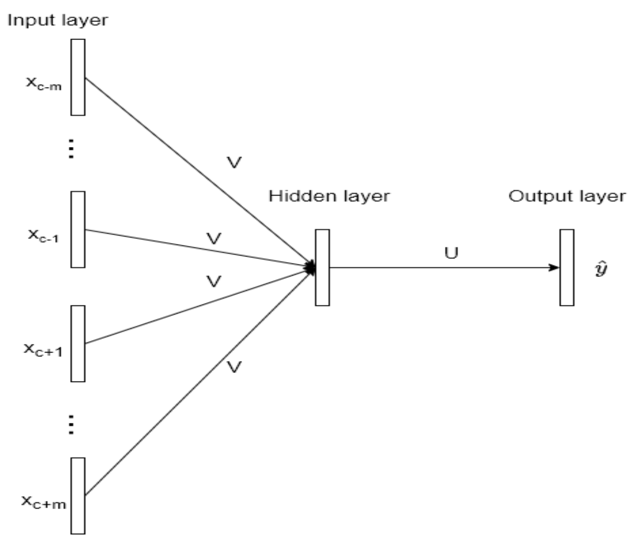


Figure 4.1: CBOW architecture

Skip-gram Model

Skip-gram [29] architecture, as shown in Figure 4.2 is similar to CBOW, but we have an inverted problem, input and output are switched. Instead of predicting the word based on the context, it tries to maximize the classification of a context based on the middle word. Increasing the range improves the quality of vectors, but also introduces additional computational complexity. More distant words are usually less related than close words, and less weight is given to them.

We use do the following notation:

- w_i: Word i from vocabulary V
- $V \in \mathbb{R}^{\mathrm{nx|V|}}$: Input word matrix
- v_i: i-th column of V, the input vector representation of word w_i
- $U \in \mathbb{R}^{\mathrm{|V|xn}}$: Output word matrix
- u_i: i-th row of U, the output vector representation of word w_i

In order to use the model, input needs to be converted into a one-hot vector x. The input vector is passed through the input word matrix V to get embedded context word vector $v_c = Vx$. This embedded vector is then passed through the output word matrix U, to get a score $z = Uv_c$. This will generate score vectors:

$$u_{c-m}, ..., u_{c-1}, u_{c+1}, ..., u_{c+m} \tag{4.8}$$

Each vector is passed through the softmax function for the probabilities:

$$\hat{y}_{c-m}, ..., \hat{y}_{c-1}, \hat{y}_{c+1}, ..., \hat{y}_{c+m} \tag{4.9}$$

These probabilities should be close to the initial one-hot vectors.

The difference here compared to CBOW is we use the Naive Bayes assumption to compute the probabilities. Given the center word, all output words

$$\begin{aligned} minimize\ \mathrm{L} &= -\log P(w_{c-m}, ..., w_{c-1}, w_{c+1}, ..., w_{c+m}|w_c) \\ &= -\log \prod_{j=0, j\neq m}^{2m} P(w_{c-m+j}|v_c) \\ &= -\log \prod_{j=0, j\neq m}^{2m} P(u_{c-m+j}|v_c) \\ &= -\log \prod_{j=0, j\neq m}^{2m} \frac{exp(u_{c-m+j}^T v_c)}{\sum_{k=1}^{|V|} u_k^T v_c} \end{aligned} \tag{4.10}$$

Note that

$$\begin{aligned} \mathrm{L} &= -\sum_{j=0, j\neq m}^{2m} \log P(u_{c-m+j}|v_c) \\ &= \sum_{j=0, j\neq m}^{2m} H(\hat{y}, y_{c-m+j}) \end{aligned} \tag{4.11}$$

where $H(\hat{y}, y_{c-m+j})$ is the cross-entropy between the probability vector $\hat{y}$ and the one-hot vector y_{c-m+j}.

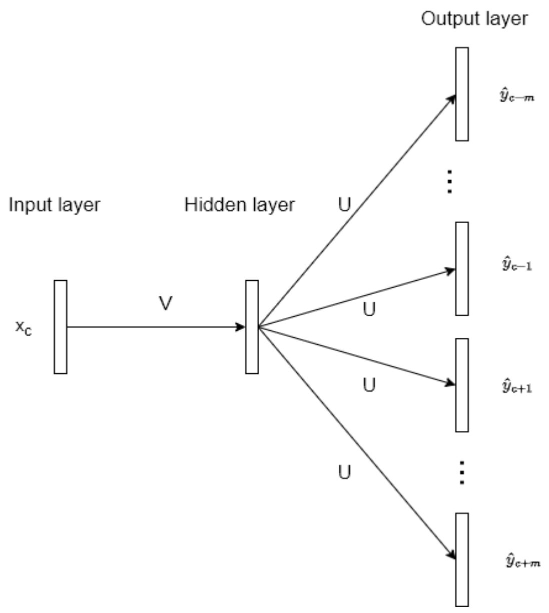


Figure 4.2: Skip-gram architecture

Word2Vec Results

Comparing the quality of different versions of word vectors is usually done to show example words and their most similar words, or relationships between pairs of vectors. For example, if we take the word small and want to find a word that will have the same relationship between them as between words biggest and big. $X = vector(biggest) - vector(big) + vector(small)$. After this vector is calculated, cosine distance is done to find the closest vector. In this case, it is the word smallest. Some more relationship founds: France is to Paris, what Germany is to Berlin.

The Word2Vec [21] model used is trained on 6B tokens from the Google News corpus. Vocabulary is restricted to 1 million most frequent words. Architecture is CBOW with a projection dimensionality of 300.

4.3.2 GloVe: Global Vectors for Word Representation

The statistics of word occurrences in a corpus are the main source of information available for unsupervised methods of learning. GloVe [22] stands for Global Vectors because the global corpus statistics are captured directly by the model. Let us first introduce the notation. Let the matrix of co-occurrences be X [30], whose entries X_{ij} show the number of times word j occurs in the context of the word i. Let $X_i = \sum_k X_{ik}$ be the number of times any word appears in the context of the word i. Let $P_{ij} = P(j|i) = \frac{X_{ij}}{X_j}$ be the probability of word j appearing in the context of the word i. To create this matrix, a single pass is needed. If the corpus is large, this can take time, but it happens only once.

The probability that word j appears in the context of the word i:

$$Q_{ij} = \frac{exp(u_j^T v_i)}{\sum_{k=1}^{|V|} exp(u_k^T v_i)} \tag{4.12}$$

Global cross-entropy loss can be calculated as

$$L = -\sum_{i \in \text{corpus}} \sum_{j \in \text{corpus(i)}} \log Q_{ij} \tag{4.13}$$

We can group words i and j that appear multiple times, and we get:

$$L = -\sum_{i=1}^{V} \sum_{j=1}^{V} X_{ij} \log Q_{ij} \tag{4.14}$$

The drawback of the cross-entropy loss is that requires the distribution Q to be properly normalized, which is an expensive summation over the entire vocabulary. Factors P and Q can be discarded [30]:

$$\hat{L} = \sum_{i=1}^{V} \sum_{j=1}^{V} X_i (\hat{P_{ij}} - \hat{Q_{ij}})^2 \tag{4.15}$$

where $\hat{P_{ij}} = X_{ij}$ and $\hat{Q_{ij}} = exp(u_j^T v_i)$ are unnormalized distributions. X_{ij} can take on large values and makes the optimization difficult. Then minimizing squared error of the logarithms of $\hat{P}$ and $\hat{Q}$:

$$\begin{aligned} \hat{L} &= \sum_{i=1}^{V} \sum_{j=1}^{V} X_i (\log(\hat{P_{ij}}) - \log(\hat{Q_{ij}}))^2 \\ &= \sum_{i=1}^{V} \sum_{j=1}^{V} X_i (u_j^T v_i - \log(X_{ij}))^2 \end{aligned} \tag{4.16}$$

The GloVe models create to take into account global statistical information. It outperforms word2vec on the analogy tasks.
The GloVe is trained on 2010 Wikipedia data with 1 billion tokens, 2014 Wikipedia data with 1.6 billion tokens, Gigaword 5 data with 4.3 tokens, and 42 billion tokens from Common Crawl [22].

4.3.3 FastText

FastText [23] represents words as the sum of the n-gram vectors. It is an extension of the continuous skip-gram model. Given a word vocabulary of size $\|V\|$, with an index of word $w \in 1, ..., \mathrm{V}$, the goal is to learn word representation for the whole vocabulary V, trying to predict well words that appear together. To define it more formally if we have words $w_1, ..., w_T$, the objective of the skip-gram model is to maximize log-likelihood:

$$\sum_{t=1}^{T} \sum_{c \in \mathrm{C_t}} \log p(w_c|w_t) \tag{4.17}$$

where the context C_t is the set of indices of context words of word w_t. The probability of a context word is the softmax:

$$p(w_c|w_t) = \frac{e^{s(w_t,w_c)}}{\sum_{j=1}^{V} e^{s}(w_t, j)} \tag{4.18}$$

where s is a scoring function of pairs (word, context) into real number scores.

We can convert the problem of predicting context words to a set of independent binary classification tasks. If a word is at position t, positive examples are all context words, while negative examples can be sampled from the dictionary. Then, for context c, with binary logistic loss, negative log-likelihood is:

$$\log(1 + e^{-s(w_t,w_c)}) + \sum_{n \in \mathrm{N_{t,c}}} \log(1 + e^{s(w_t,n)}) \tag{4.19}$$

where $N_{t,c}$ is a set of negative examples sampled. If we use notation for logistic loss: $l : x \rightarrow log(1 + e^{-x})$, then:

$$\sum_{t=1}^{T}\left[\sum_{c\in \mathrm{C_t}} l(s(w_t,w_c)) + \sum_{n\in \mathrm{N_{t,c}}} l(-s(w_t,n))\right] \tag{4.20}$$

Vectors v_w and v_w are referred as input and output vectors from $\mathbb{R}^n$. The score s thus can be computed as $s(w_t, w_c) = u_{w_t}^T v_{w_c}$, which is skip-gram model.

Every word w is represented as a bag of character n-gram. Special symbols $<$ and $>$ are added at the beginning and end of words. For example if we take word *where* and $n = 3$, then the character n-grams are: <wh, whe, her, ere, re>, with special sequence <where>.
For a given dictionary of size G and word w, let $G_w \subset 1,,,,G$ be the set of n-grams appearing in w. Then, the word can be represented by the sum of the vector representations of its n-grams.

$$s(w,c) = \sum_{g\in \mathrm{G_w}} \mathbf{z_g^T v_c} \tag{4.21}$$

This model allows sharing of the representations across words.
FastText is trained on Wikipedia data from nine languages: Arabic, Czech, German, English, Spanish, French, Italian, Romanian and Russian [23].

4.4 Custom Word Embeddings

Instead of using pre-trained word embeddings we can either use our vocabulary and built task-specific embeddings such as Word2Vec, GloVe, and FastText. These all are part of unsupervised learning. Instead of unsupervised learning, we can use supervised learning and learn dense vector representation. PyTorch offers an Embedding layer from the nn module. Given the vocabulary size of N most common words, and the dimensionality of embedding, a dense layer is created with the input index of a word in a vocabulary. During the classification task, this dense layer is found during model optimization. This way, we have vocabulary created and trained for a specific task. The downside of this approach is that all pre-trained word embeddings focus solely on finding the best vectorization, while this is only the side output of the optimization of the overall model.

4.5 Contextual Embeddings

So far, all previous embeddings had only one encoding for each word, even though that does not want we always want. Take for example homonyms. They are the same words, but with different meanings. None of the above mentions vectorizations did not take into account that. Transformer models [14] in deep learning are the first model to have different vectorization for different words. We will focus here on the Bidirectional Encoder Representation - BERT [15].
BERT embeddings take as an input the whole sentences, and tokenization is done with a BERT tokenizer, which splits sentences into tokens, where tokens can be also only parts of the word. For example, the word 'playing' is tokenized into 'play' and '###ing'. We refer to this as token embeddings. Each sentence has its own unique number, which is used as segment embedding. Finally, each position of the token is embedded as positional embedding, from the sinus/cosines function of the index of position. Adding token embeddings, sentence (segment) embeddings, and position embeddings, we get final embeddings. This is shown in Figure 4.3.

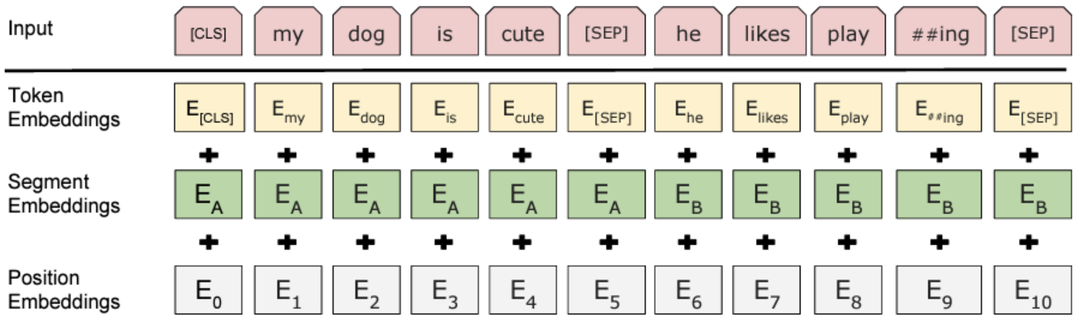

Input
[CLS]
my
dog
is
cute
[SEP]
he
likes
play
##ing
[SEP]
Token Embeddings
$E_{[CLS]}$ E_{my} E_{dog} E_{is} E_{cute} $E_{[SEP]}$ E_{he} E_{likes} E_{play} $E_{\#\#ing}$ $E_{[SEP]}$
Segment Embeddings
E_A E_A E_A E_A E_A E_A E_B E_B E_B E_B E_B
Position Embeddings
E_0 E_1 E_2 E_3 E_4 E_5 E_6 E_7 E_8 E_9 E_{10}

Figure 4.3: Bert input embeddings [15]

Chapter 5

Machine Learning Concepts

5.1 Bagging

Suppose we fit a model to our training data $\mathbf{Z} = (x_1, y_1, ..., (x_N, y_N))$, obtaining prediction $\hat{f}(x)$ at input x. Bootstrap aggregation or bagging [31] calculates the average from all predictions of bootstrap samples. This way, variance is reduced. For each bootstrap sample $\mathbf{Z^{*b}}, b = 1, 2, ..., B$, model is fitted to obtain prediction $\hat{f}^{*b}(x)$. The estimate defined by bagging is defined as:

$$\hat{f}_{bag}(x) = \frac{1}{B} \sum_{b=1} B \hat{f}^{*b} \tag{5.1}$$

Let $\hat{P}$ be the empirical distribution with equal probability $\frac{1}{N}$ of choosing data point (x_i, y_i). The "true" bagging estimate is defined by $E_{\hat{P}} \hat{f}^*(x)$, where $\mathbf{Z}^* = (x_1^*, y_1^*), ..., (x_N^*, y_N^*)$. This is a Monte Carlo estimate of the true bagging estimate, approaching it as $B \to \infty$.

Let us suppose that tree creates a classifier $\hat{G}(x)$ with K classes. Consider vector function $\hat{f}(x)$ with one one value and rest $K - 1$ are zeros, such that $\hat{G}(x) = \text{argmax}_k \hat{f}(x)$. The bagged estimate $\hat{f}_{bag}(x)$ is a K-vector $[p_1(x), p_2(x), ..., p_K(x)]$, with $p_k(x)$ equal to the proportion of trees predicting class k at x. The bagged classifier selects the class with the most "votes" from the B trees, $\hat{G}_{bag}(x) = \text{argmax}_k \hat{f}_{bag}(x)$.

5.2 Metrics

To know which model performs better, we need to define metrics that will be indicators of how each model performs. In classification tasks, the most common metrics are confusion matrix, accuracy, precision, and recall [32].

The confusion matrix in Table 5.1 represents the table of model predictions for each class.

		True sentiment		
		Positive	Negative	Total
Predicted sentiment	Positive	TP	FP	$TP + FP$
	Negative	FN	TN	$FN + TN$
	Total	$TP + FN$	$FP + TN$	N

Table 5.1: Confusion Matrix
[33]

True positive is the number of positive samples that the model correctly classified (TP), false negative is the number of positive classes that which model classified as negative (FN), false positive is the number of negative classes that which model classified as positive (FP) and true negative is the number of negative samples which model correctly classified (TN).

Accuracy is the sample metric for evaluating classification models. It is defined as the number of correct predictions divided by the total number of predictions, or:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{5.2}$$

The recall is defined as the proportion of detected positive class with all positive class instances, or:

$$\text{Recall} = \frac{TP}{TP + FN} \tag{5.3}$$

The precision is defined as the proportion of detected positive class with all positive class predictions, or:

$$\text{Precision} = \frac{TP}{TP + FP} \tag{5.4}$$

The F1 score is defined as a weighted average between precision and recall (harmonic mean), or:

$$\text{F1} = \frac{2}{\frac{1}{recall} + \frac{1}{precision}} \tag{5.5}$$

Precision and recall are here shown for the positive class, but they can be also calculated for the negative class, in the same way.

Let the confusion matrix for $n = 5$ classes $(C_1, C_2, C_3, C_4, C_5)$ be as defined in Table 5.2, where columns are actual classes, and rows are predicted, classes. Entry C_{ij} in the matrix represents the number of elements from class j classified as class i. There are two types to calculate precision and recall, micro and macro. In micro metrics, the binary classification matrix is created as one-vs-all, where we treat the class of interest C_i as a positive class and all others as negatives, and a sum of their values are combined (as shown in binary classification metrics). In macro metrics, scores are calculated for each class individually, and then global metrics are defined as the unweighted mean of the measures. Macro scores, from confusion matrix defined in Table 5.2, are calculated as follows:

$$\begin{aligned}
TP_i &= C_{ii} \\
FP_i &= \sum_{l=1}^{5} C_{il} - TP_i \\
FN_i &= \sum_{l=1}^{5} C_{li} - TP_i \\
FN_i &= \sum_{l=1}^{5} \sum_{k=1}^{5} C_{kl} - TP_i - FP_i - FN_i
\end{aligned}$$

		Actual Classes				
		C_1	C_2	C_3	C_4	C_5
Predicted Classes	C_1	C_{11}	C_{12}	C_{13}	C_{14}	C_{15}
	C_2	C_{21}	C_{22}	C_{23}	C_{24}	C_{25}
	C_3	C_{21}	C_{32}	C_{33}	C_{34}	C_{35}
	C_4	C_{41}	C_{42}	C_{43}	C_{44}	C_{45}
	C_5	C_{51}	C_{52}	C_{53}	C_{54}	C_{55}

Table 5.2: Confusion Matrix for 5-classes [33]

5.3 Cross Validation

Cross validation [34] is used for making results precise and robust, because in one random split of the data, we may get really great results, which is not realistic, but if that is repeated k-times, then we feel more confident in the final metrics. The more splits are done, the more robust results are. One of the common techniques in cross validation is K-fold validation.
In K-fold validation, as shown in Figure 5.1, data is split into K folds, as in name suggests. From K-folds, $K-1$ fold is used for training, while K-th fold is used for validation, and this procedure is repeated K-times. Every data point is only once used in the test set and $K-1$ times in the training set.

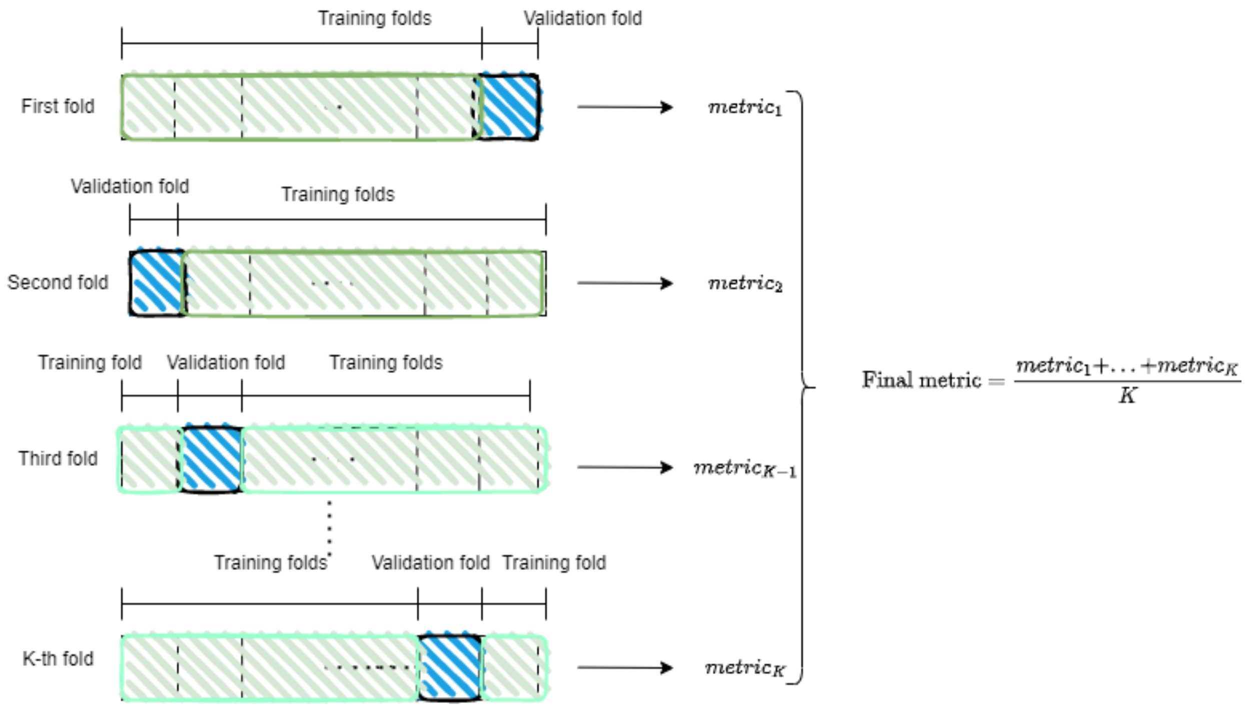


Figure 5.1: K-cross validation

Chapter 6

Machine Learning Models

6.1 Naive Bayes Classifier

Let d be a document, c true class for the document, and C set of all classes. Naive Bayes [35] is a probabilistic classifier, which means that for a given document d it outputs class probabilities and gives the class $\hat{c}$ with the maximum posterior probability.

$$\hat{c} = \underset{c \in C}{\operatorname{argmax}}\, p(c|d) \tag{6.1}$$

This is known as **Bayesian inference** and it is applied to text classification, amongst others. Using Bayes‘ rule, this is transformed to:

$$\hat{c} = \underset{c \in C}{\operatorname{argmax}}\, p(c|d) = \underset{c \in C}{\operatorname{argmax}}\, \frac{p(d|c)p(c)}{p(d)} \tag{6.2}$$

We used the property that all classes are divided by $p(d)$, therefore we can disregard it, which means that

$$\frac{p(d|c)p(c)}{p(d)} \rightarrow p(d|c)p(c) \tag{6.3}$$

Now, using (6.3), the output class can be calculated as:

$$\hat{c} = \underset{c \in C}{\operatorname{argmax}}\, p(c|d) = \underset{c \in C}{\operatorname{argmax}}\, p(d|c)p(c) \tag{6.4}$$

The most probable class $\hat{c}$ is calculated as the product of prior probability $p(c)$ and the likelihood of $p(d|c)$.

Let the document d has a set of features $f_1, ..., f_n$:

$$\hat{c} = \operatorname*{argmax}_{c \in C} p(f_1, f_2, ..., f_n|c)p(c) \tag{6.5}$$

where $p(f_1, f_2, ..., f_n|c)$ is likelihood, and $p(c)$ is prior.

Even with these modifications, is still hard to compute the probability, estimating the probability of every possible combination of features would require huge numbers of parameters and a large dataset. Naive Bayes, therefore, makes some assumptions.
One of the assumptions is the bag of word assumption: the position of the word does not matter and has the same effect whether is the first, second, or last word in a document. So features $f_1, f_2, ..., f_n$ only encode word, not position.
The second assumption is the conditional independence assumption that the probabilities $p(f_i|c)$ are independent given the class c and can be calculated as follows:

$$p(f_1, f_2, ..., f_n|C) = p(f_1|c)p(f_2|c)...p(f_n|c) \tag{6.6}$$

The final equation can be rewritten as:

$$c_{NB} = \operatorname*{argmax}_{c \in C} p(c) \prod_{f \in F} p(f|c) \tag{6.7}$$

In order to apply Naive Bayes, we go through each word in the document:

$$c_{NB} = \operatorname*{argmax}_{c \in C} p(c) \prod_{i \in \text{positions}} p(w_i|c) \tag{6.8}$$

Naive Bayes calculations are done in log space, to avoid underflow and increase speed. Thus:

$$c_{NB} = \operatorname*{argmax}_{c \in C} \log p(c) + \sum_{i \in positions} \log p(w_i|c) \tag{6.9}$$

6.1.1 Training the Naive Bayes Classifier

Let N_c be the number of documents in our training data with class c and N_{doc} be the number of documents in the corpus. Then:

$$\hat{p}(c) = \frac{N_c}{N_{doc}} \tag{6.10}$$

Going back to (6.6) and to get probability $p(f_i|c)$, the existence of a word in documents' bag of words is a feature, and let that be $p(w_i|c)$. This probability is calculated as the number of times the word w_i appears from all words in all documents in topic c.

$$\hat{p}(w_i|c) = \frac{count(w_i, c)}{\sum\limits_{w \in W} count(w, c)} \tag{6.11}$$

There is only one problem here with maximum likelihood training. If we try to estimate some word for a specific class, but there are no examples in training documents about that word and that class. In such a case, the probability will be 0. But since naive Bayes naively multiplies all the feature likelihoods together, zero probabilities in the likelihood term for any class will cause the probability of the class to be zero.
One solution is add-one (Laplace) smoothing.

$$\hat{p}(w_i|c) = \frac{count(w_i, c) + 1}{\sum\limits_{w \in W} (count(w, c) + 1)} = \frac{count(w_i, c) + 1}{(\sum\limits_{w \in W} count(w, c)) + |V|} \tag{6.12}$$

If an unknown word is appearing in test data, then the solution is to remove them. Stop words like *a*, *the*, and so on, appear very frequently without bringing many contexts, so it is usually the best idea to ignore them.

6.2 Logistic Regression

Given predictor G, and input vector x, G can separate input space with decision boundaries, giving all vectors from each boundary the same output. Let us assume that there are K classes, and fitted linear model for the k-th class is $\hat{f}_k(x) = \hat{\beta}_{k0} + \hat{\beta}_k^T x$. The decision boundary between class k and l is that set of points for which $\hat{f}_k(x) = \hat{f}_l(x)$, that is set $\{x : (\hat{\beta}_{k0} - \hat{\beta}_{lo}) + (\hat{\beta}_k - \hat{\beta}_l)^T x = 0\}$, an affine set or hyperplane. Input space is divided with piece-wise hyperplane decision boundaries.

The decision boundary [31] is the set of points for which the log odds are zero, and this is a hyperplane defined by $\{x|\beta_0 + \beta^T = 0\}$

For K classes, the logistic regression gives posterior probabilities with linear functions, making sure that the sum of probabilities for all classes is 1, and all class probabilities stay between $[0, 1]$. Logistic regression has form:

$$\begin{aligned}
\log\left(\frac{p(G=1|X=x)}{p(G=K|X=x)}\right) &= \beta_{10} + \beta_1^T x \\
\log\left(\frac{p(G=2|X=x)}{p(G=K|X=x)}\right) &= \beta_{20} + \beta_2^T x \\
&. \\
&. \\
&. \\
\log\left(\frac{p(G=K|X=x)}{p(G=K|X=x)}\right) &= \beta_{(K-1)0} + \beta_{K-1}^T x
\end{aligned} \tag{6.13}$$

The model uses $K-1$ log odds or logit transformations. If we the drop last class as the denominator:

$$p(G=1|X=x) = \frac{exp(\beta_{k0} + \beta_k^T x)}{1 + \sum_{l=1}^{K-1} exp(\beta_{l0} + \beta_l^T x)}, k = 1, ..., K-1 \tag{6.14}$$

$$p(G=K|X=x) = \frac{1}{1 + \sum_{l=1}^{K-1} exp(\beta_{l0} + \beta_l^T x)} \tag{6.15}$$

Let the entire parameter set $\theta = (\beta_{10}, \beta_1^T, ..., \beta_{K-1}^T)$. Let $p(G = k|X = x) = p_k(x; \theta)$. If $K = 2$, there is only one linear function.

6.2.1 Training Logistic Regression

Logistic regression [31] is fit with maximum likelihood using conditional likelihood G given X. The log-likelihood for N observation is

$$l(\theta) = \sum_{i=1}^{N} \log p_{g_i}(x_i; \theta) \tag{6.16}$$

where $p_k(x_i; \theta) = p(G = k|X = x_i; \theta)$. In case of two classes we can write $p_1(x; \theta) = p(x; \theta)$ and $p_2(x_i; \theta) = 1 - p(x_i; \theta)$. Log-likelihood now becomes:

$$\begin{aligned} l(\beta) &= \sum_{i=1}^{N} \{y_i \log p(x_i;\beta) + (1-y_i)\log(1-p(x_i;\beta))\} \\ &= \sum_{i=1}^{N} \{y_i \beta^T x_i - \log(1+e^{\beta^T x_i})\} \end{aligned} \tag{6.17}$$

where $\beta = \{\beta_{10}, \beta_1\}$, and vector x_i includes constant term 1 to accommodate the intercept.
Now, to maximize log-likelihood, we take the derivative and set it to zero.

$$\frac{\partial l(\beta)}{\partial \beta} = \sum_{i=1}^{N} x_i(y_i - p(x_i;\beta)) = 0 \tag{6.18}$$

This is a system of $p+1$ non-linear equations in β. Because in x_i first component is 1, then $\sum_{i=1}^{N} y_i = \sum_{i=1}^{N} p(x_i;b)$. To solve the previous equation, we use the Newton–Raphson which requires the Hessian matrix (second derivative):

$$\frac{\partial^2 l(\beta)}{\partial\beta\,\partial\beta^T} = -\sum_{i=1}^{N} x_i x_i^T p(x_i;\beta)(1-p(x_i;\beta)) \tag{6.19}$$

Starting with β^{old}, a single Newton update is and derivatives are evaluated at $\beta^o ld$:

$$\beta^{new} = \beta^{old} - \left(\frac{\partial^2 l(\beta)}{\partial\beta\,\partial\beta^T}\right)^{-1} \frac{\partial l(\beta)}{\partial\beta} \tag{6.20}$$

Let y be the vector of y_i, X the $N \times (p+1)$ matrix of x_i values, and p the vector of fitted probabilities with with i-th element $p(x_i;\beta^{old})$ and W a $N \times N$ diagonal matrix of weights with i-th diagonal element $p(x_i;\beta^{old})(1-p(x_i;\beta^{old}))$. Then:

$$\frac{\partial l(\beta)}{\partial\beta} = X^T(y-p) \tag{6.21}$$

$$\frac{\partial l(\beta)}{\partial\beta\beta^T} = -X^T W X \tag{6.22}$$

The Newton step is [31]:

$$\begin{aligned}\beta^{new} &= \beta^{old} + \left(X^T W X\right)^{-1} X^T(y-p) = \\ &= \left(X^T W X\right)^{-1} X^T W (X\beta^{old} + W^{-1}(y-p)) = \\ &= \left(X^T W X\right)^{-1} X^T W z\end{aligned} \tag{6.23}$$

Newton's step has been re-expressed as a weighted least squares step with the response:

$$z = X\beta^{old} + W^{-1}(y-p) \tag{6.24}$$

which is known also as an adjusted response. These equations get solved repeatedly since at each iteration p changes, and hence so do W and z. This algorithm is known as iteratively re-weighted least squares or IRLS since each iteration solves the weighted least square problem:

$$\beta^{new} \leftarrow \underset{\beta}{\operatorname{argmin}} (z - X\beta)^T W (z - X\beta) \tag{6.25}$$

It looks like $\beta = 0$ is a good starting value for the iterative procedure, although convergence is never guaranteed. Typically the algorithm does converge since the log-likelihood is concave, but overshooting can occur. In the rare cases that the log-likelihood decreases, step size halving will guarantee convergence [31].

6.3 Decision Tree

A decision tree [36], as shown in Figure 6.1, is one of the most intuitive models for classification. It partitions input space and to each subspace gives a label. They are easy to use and easy to interpret. Decision trees are non-parametric. models: there are no weight parameters. It can be used both in for numerical and categorical attributes. Typically, data comes in tabular form Table 6.1.

Data point	Feature 1	...	Feature K	Label
Data point 1	feature 1	...	feature k	label
Data point 2	feature 1	...	feature k	label

Table 6.1: Example of tabular data

[37]

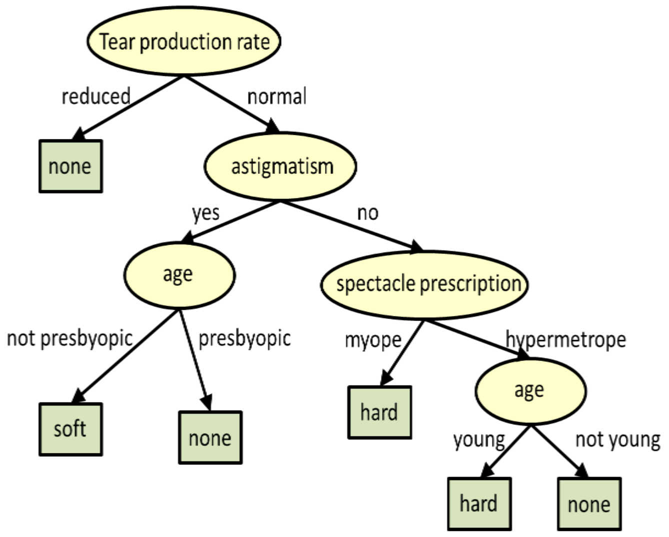


Figure 6.1: Example of decision tree [37]

Let us set up the notation. Let x be a data point with d features, $A_1, ..., A_d$ and class y, and set of all training data is X. Let S be the partitioning rule which splits the training dataset X into $X_1, .., X_k$. A decision tree is a rooted tree in which each set of children of each parent node corresponds to a partitioning (X_S) of the parent's data set, with the full data set associated with the root. The number of items in X_i that belong to class y_j is $|X_{ij}|$. The probability that a randomly selected member of X_i is of class y_j is $p_{ij} = |X_{ij}|$.

6.3.1 Node Splitting

During the splitting of a tree, one should think about which split is the best. There are a couple of different possible splits [36]:

- Binary attributes: Only one split is possible.
- Categorical attributes: If attribute A is unordered, then the domain of A is a mathematical set. Any nonempty proper subset S of A defines a binary split $S, A\backslash S$. After ruling out redundant and empty partitions, we have $2^{(k-1)} - 1$ possible binary splits. However, some algorithms make k-way splits instead.
- Numerical attributes: If there are k different values, then we can make either (k-1) different binary splits or one single k-way split.

Algorithm 1 Decision tree algorithm [37]

```
1: Assign all training instances to the root of the tree. Set the current node
   to the root node.
2: For each attribute
      • Partition all data instances at the node by the value of the attribute.
      • Compute the information gain ratio from the partitioning.
3: Identify a feature that results in the greatest information gain ratio. Set
   this feature to be the splitting criterion at the current node.
4: If the best information gain ratio is 0, tag the current node as a leaf and
   return.
5: Partition all instances according to the attribute value of the best feature.
6: Denote each partition as a child node of the current node.
7: For each child node:
      • If the child node is "pure" (has instances from only one class) tag it
        as a leaf and return.
      • If not, set the child node as the current node and go to step 2.
```

We need somehow to measure the goodness of the splits. An ideal scenario would be that each split contains only one class. To optimize the decision tree, we want a splitting rule that minimizes the impurity function [36].

Definition 1. *An impurity function F for an m-state discrete variable Y is a function defined on the set of all m-tuple discrete probability vectors $(p_1, p_2, ..., p_m)$ such that:*

1. *F is maximum only at* $(\frac{1}{m}, \frac{1}{m}, ..., \frac{1}{m})$
2. *F is minimum only at the "purity points"* $(1, 0, ..., 0), (0, 1, 0..., 0), ..., (0, 0, ..., 0)$
3. *F is symmetric with respect to* $p_1, p_2, ..., p_m$

6.3.2 Impurity Functions

The following impurity functions [36] are commonly used when training decision tree models:

1. Error Rate
   This is a measure of misclassified items and it is the most simple one. If y_j is the class that makes the majority of all classes in split, then the error rate for X_i is $E(X_i) = \frac{|y \neq y_j : (x,y) \in X_i|}{|X_i} = 1 - p_{ij}$. The error rate for the entire split X_S is the weighted sum of the error rates for each subset. This equals the total number of misclassifications, normalized by the size of X.

$$\Delta F_{error}(S) = E(X) - \sum_{i \in S} \frac{X_i}{X} E(X_i) \tag{6.26}$$

2. Entropy and Information Gain
   Information entropy is defined as the degree of uncertainty of a (discrete) random variable Y

$$H_X(Y) = -\sum_{y \in Y} p_y \log(p_y) \tag{6.27}$$

   Information entropy can be thought of as the expected amount of information, needed to describe the state of a system. The purer the system is, the less information is required. If the system has all objects in the same state, then the entropy is 0. In contrast, the more equal the distribution of classes is, the higher entropy is. If we have a system with split criteria, If the system is pre-partitioned into subsets according to some splitting rule S, then the information entropy of the overall system is the weighted sum of the entropies for each partition, $H_{X_i}(Y)$. This is equivalent to the conditional entropy $H_X(Y|S)$

$$\begin{aligned}\Delta F_{infoGain}(S) &= -\sum_{y \in Y} p_y \log p_y + \sum_{i \in S} \frac{|X_i|}{|X|} \sum_{y \in Y} p_{iy} \log p_{iy} = \\ &= H_X(Y) - \sum_{i \in S} H_{X_i}(Y) = \\ &= H_X(Y) - H_X(Y|S)\end{aligned} \tag{6.28}$$

   A shortcoming of the information gain criterion is that it is biased towards splits with larger k. Given a candidate split, if subdividing any subset provides additional class differentiation, then the information gain score will always be better.

3. Gini Criterion (CART)
   The Gini index is used as a splitting criterion. This can be treated as an expected error of splitting criteria as opposed to randomly classifying the algorithm.

$$Gini(X_i) = \sum_{y \in Y} p_{iy}(1 - p_{iy}) = 1 - \sum_{y \in Y} p_{ij}^2 \tag{6.29}$$

$$\Delta F_{Gini}(S) = Gini(X) - \sum_{i \in S} \frac{|X_i|}{|X|} Gini(X_i) \tag{6.30}$$

   The Gini index is biased towards splits with larger k. [37]

Based on the splitting rules, the decision tree starts splitting recursively, as long as there is a decrease in the error function, but this can cause the model to overfit. If there are no hard limits in the depth of a tree, the model can learn exact splitting criteria to achieve a perfect score on the training set, which is not realistic. To reduce overfitting, tree pruning, as in Figure 6.2, techniques are used to get a more shallow tree with slightly less accuracy.

1. Cost Complexity Pruning
   In cost complexity pruning, sometimes also called error complexity pruning, idea is to replace a subtree with a single node. If L_X is the set of leaf node data subset of X, then:

$$error\_complexity = \frac{E(X) - \sum_{L_i \in L_X} E(L_i)}{|L_X| - 1} \tag{6.31}$$

   For each node, compute the complexity error, a, and choose the one with the smallest value. Compute error complexity for each internal node, and convert the one with the smallest value (least increase in error per leaf) to a leaf node. Based on the test data error, with the smallest standard error, the best tree is chosen.

2. Critical Value Pruning
   The idea is similar to the previous approach, except that measure for pruning is the same as for the growing tree. If all the splitting criteria values in nodes are below some threshold, then the subtree is replaced with a single leaf value.

Once a decision tree is learned, it can be used to evaluate new instances to determine their class. The instance is passed down the tree, from the root, until it arrives at a leaf. The class assigned to the instance is the class for the leaf. [36]
In the Figure 6.2 we have pruned the decision tree. In comparison to the original tree Figure 6.1, we can see it has fewer subtrees and it is more shallow.

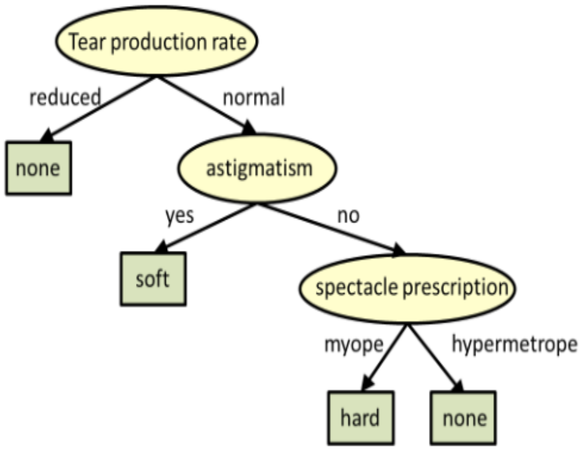


Figure 6.2: Example of pruned decision tree [37]

6.4 Random Forest

Random forest is based on bagging or bootstrap aggregation [31]. Bagging seems to work, especially well for high-variance, low-bias procedures, such as trees. Random forest is a modification of bagging that builds a large collection of de-correlated trees and then averages them.

6.4.1 Definition of Random Forest

The idea in bagging is to deal with many noisy unbiased models, and reduce variance. Trees are good for bagging since they can have a low bias and are able to capture the relationship between features and output class. They

can be noisy, so by averaging them one can get better predictions. A tree expectation is the same as the average expectation of trees.
For an average tree B, created from independent and identically distributed trees, where each tree has variance σ^2, has a variance $\frac{1}{B}\sigma^2$.
Before creating trees, only a subset of instances of data is used for the training.

Algorithm 2 Random Forest Algorithm [31]

```
1. For b=1 to B:
    (a) Draw a bootstrap sample $Z*$ of size $N$ from the training data.
    (b) Grow a decision tree $T_b$ to the bootstrapped data, by recursively
        repeating the following steps for each terminal node of the tree,
        until the minimum node size $n_{min}$ is reached.
          i. Select m variables at random from the p variables.
         ii. Pick the best variable/split-point among them.
        iii. Split the node into two daughter nodes.
2. Output the ensemble of Trees ${T_b}_1^B$ To make a prediction for a new point
   let $\hat{C}_b$ be the class prediction of the $b$-th random forest tree. Then $\hat{C}_{rf}^B$
   = majority vote $\hat{C}_b(x)_1^B$.
```

6.5 Support Vector Machines

6.5.1 Optimal Separating Hyperplanes

Let us look at the optimization problem [**cite {statistical_learning}**]:

$$\begin{gathered}\max_{\beta,\beta_0,\|\beta\|} M \\ \text{subject to } y_i(x_i^T\beta + \beta_0 = 1) \geq M, i = 1, ..., M\end{gathered} \tag{6.32}$$

Here, constraints in optimization make sure that all points are at least M distance away from the decision boundary defined by β and β_0. M is chosen to be the largest and to satisfy all constraints. We can replace constraint $\|\beta\| = 1$ by changing it with:

$$\frac{1}{\|\beta\|} y_i(x_i^T \beta + \beta_0) \geq M \tag{6.33}$$

or equivalently:

$$y_i(x_i^T \beta + \beta_0) \geq M\|\beta\| \tag{6.34}$$

We can set $\|\beta\| = \frac{1}{M}$, getting (6.32) reformed optimization problem:

$$\begin{gathered} \min_{\beta,\beta_0} \frac{1}{2}\|\beta\|^2 \\ \text{subject to } y_i(x_i^T \beta + \beta_0) \geq 1, i = 1, ..., N \end{gathered} \tag{6.35}$$

Margin is of thickness $\frac{1}{\|\beta\|}$. We choose β and β_0 such that thickness is maximized. This is a convex optimization problem. The Lagrange function, to be minimized with respect to β and β_0 is:

$$L_p = \frac{2}{\|\beta\|^2} - \sum_{i=1}^{N} \alpha_i [y_i(x_i^T \beta + \beta_0) - 1] \tag{6.36}$$

Taking derivatives and setting them to zero:

$$\beta = \sum_{i=1}^{N} \alpha y_i x_i \tag{6.37}$$

$$0 = \sum_{i=1}^{N} \alpha_i y_i \tag{6.38}$$

If the substitution is made to (6.35), we have Wolfe's dual

$$\begin{gathered} L_D = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{k=1}^{N} \alpha_i \alpha_k y_i y_k x_i^T x_k \\ \text{subject to } \alpha_i \geq 0 \text{ and } \sum_{i=1}^{N} \alpha_i y_i = 0 \end{gathered} \tag{6.39}$$

The solution is found by maximizing L_D in the positive orthant. The solution must satisfy Karush-Kuhn-Tucker conditions, (6.37), (6.38), (6.39):

$$\alpha_i[y_i(x_i^T\beta + \beta_0) - 1] = 0, \forall i \tag{6.40}$$

We can observe that:

- if $\alpha_i > 0$, then $y_i(x_i^T\beta + \beta_0)$, x_i is on the boundary
- if $y_i(x_i^T\beta + \beta_0) > 1$, x_i is not on the boundary and $\alpha_i = 0$

From (6.37), vector β is defined as a linear combination of x_i, or support vectors x_i. Those are the points that define boundaries.
The optimal separating hyperplane is a function $\hat{f}(x) = x^T\hat{\beta} + \hat{\beta}_0$ for making new predictions:

$$\hat{G}(x) = \text{sign } \hat{f}(x) \tag{6.41}$$

Some test examples may fall inside of the separating boundary, that's why it is important to have it as thick as possible margin.

In the Figure 6.3, we have green and red points from different classes, and blue points, where two blue points closer to red points belong to the red class, while a blue point close to green points belongs to the green class. Blue points represent support vectors and they are used to create a hyperplane between them. The thick yellow color indicates the width of the margin.

6.5.2 SVM for classification

Let the training data consists of N pairs $\{(x_1, y_1), (x_2, y_2), ..., (x_N, y_N)\}$ with $x_i \in \mathbb{R}^p$ and $y_i \in \{-1, 1\}$ are classes. Let the hyperplane be defined by:

$$\{x : f(x) = x^T\beta + \beta_0 = 0\} \tag{6.42}$$

where β is a unit vector $\|\beta\| = 1$. Then, the rule for the classification is:

$$G(x) = \text{sign}[x^T\beta + \beta_0] \tag{6.43}$$

The distance from point x to the hyperplane is given by function f with $f(x) = x^T\beta + \beta_0$. If classes are linearly separable, we can find a function $f(x) = x^T\beta + \beta_0$ with $y_i f(x_i) > 0$, $\forall i$. For linearly separable problems, hyperplane can be found with the biggest margin between training points.

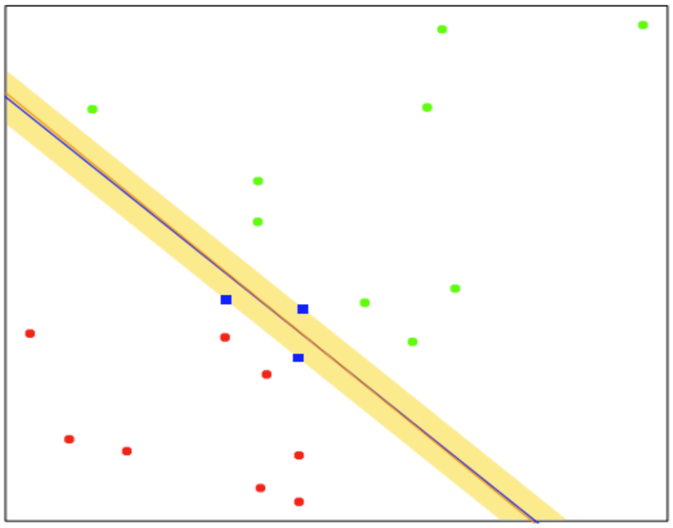


Figure 6.3: Example of SVM with separable classes [31]

The optimization defined in (6.32) describes this problem, and it can be transformed into (6.35).

If data points are not linearly separable, then there is no perfect solution [31], but still, a hyperplane can be found to maximize M and tries to classify as many as possible data points correctly. If we define slack variables $\xi = (\xi_1, ..., \xi_N)$, then constraints can be modified as followed:

$$y_i(x_i^T\beta + \beta_0) \geq M - \xi_i \tag{6.44}$$

Or

$$y_i(x_i^T\beta + \beta_0) \geq M(1 - \xi_i) \tag{6.45}$$

$\forall i, \xi_i \geq 0, \sum_{i=1}^{N} \xi_i \leq$ constant.

Two different choices lead to two different solutions. The first one is a nonconvex problem and measures the overlap in actual distance from the margin, while the second is convex and measures the overlap in the relative distance, which changes with width margin M width.

In the constrain $y_i(x_i^T\beta + \beta_0)$, the value ξ_i is proportional amount by which the prediction $f(x_i) = x_i^T\beta + \beta_0$ is on the wrong side of the margin. If ξ_i, then misclassification occurs. Bounding $\sum \xi$ by some value K sets a limit to the number of wrongly classified samples.

We can define $M = \frac{1}{\|\beta\|}$, and write:

$$\min\|\beta\| \text{ subject to } \begin{cases} y_i(x_i^T\beta + \beta_0) \geq 1 - \xi_i, \forall i \\ \xi_i \geq 0, \sum \xi_i \leq \text{ constant} \end{cases} \tag{6.46}$$

Using Lagrange multipliers and KKT conditions, support vectors are found.

In Figure 6.4, we have two examples of SVM classification. In the image on the left, data is separable, while on the right image is not. In both cases, a hyperplane is defined with equation $x^T\beta + \beta_0 = 0$. The yellow area is margin width and it is defined with $2M$, where $M = \frac{1}{\|\beta\|}$. The points ξ_i are on the wrong side of the hyperplane by amount $\xi_i^* = M\xi_i$, where points on the correct side have $\xi_i = 0$. The total distance of points on the wrong side is $\sum_i \xi_i$.

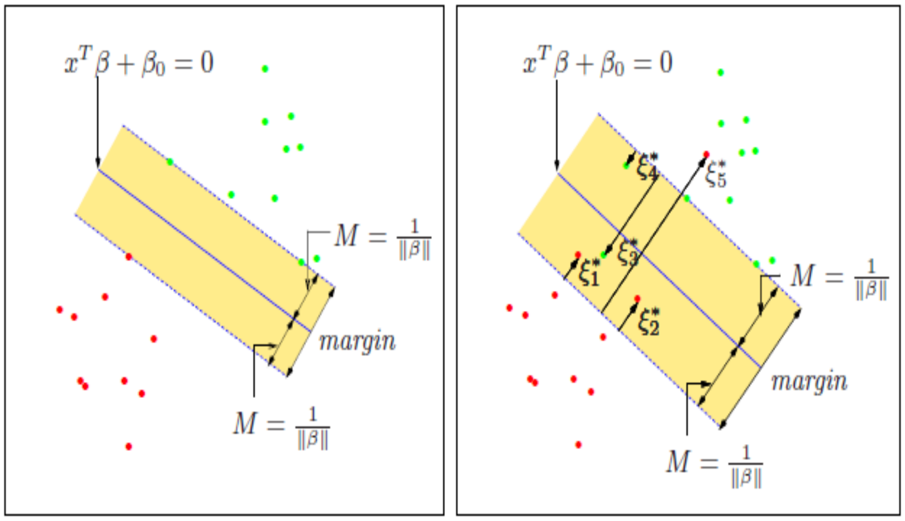


Figure 6.4: SVM classifier [31]

Sometimes, not all data points are linearly separable, and it is desirable to map input space into some feature space where classes will be linearly separable. This is known as the kernel trick. The kernel does not map directly data points into new feature space, rather it maps pair of vectors into a dot product. This way, SVM becomes a more powerful tool for classification tasks.

Chapter 7

Deep Learning models

7.1 ANN

Artificial neural networks were inspired by the brain cells, as in Figure 7.1, and how they send information. If there is enough excitement between cells, over some threshold, information from one cell is passed to the next cell. This was used as the initial logic for making artificial neural networks.

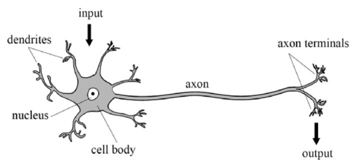


Figure 7.1: Example of a biological neuron [38]

7.1.1 Perceptron

The most simplified neural network is perception [39], as in Figure 7.2. Perceptron is defined as:

$$y(x) = f(w^T g(x)) \tag{7.1}$$

and f is a step function

$$f(a) = \begin{cases} +1, & a \leq 0 \\ -1, & a \geq 0 \end{cases} \tag{7.2}$$

Vector $g(x)$ contains bias component $g_0(x) = 1$. In perceptron, it is convenient to write output classes as -1 and 1, instead of 0 and 1. Learning parameters w can be found using error function minimization. This is not easy, because the error is a piece-size constant function of w. Changing w based on the gradients cannot be applied.

Therefore a different function needs to be used for perceptron. Vector w should satisfy that $x^T g(x_n) > 0$, if x_n is in class C_1, otherwise $x^T g(x_n) < 0$, if x_n is in class C_2. This criterion is defined as:

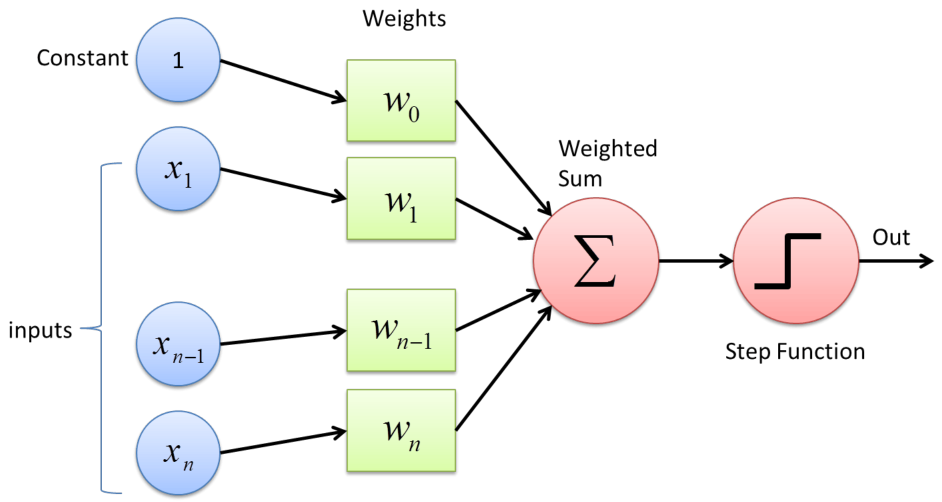


Figure 7.2: Example of a perceptron [40]

$$E_P(w) = -\sum_{n \in M} w^T g_n t_n \tag{7.3}$$

where M is set of all misclassified patterns and t_n is class value. Only the misclassified item contributes to the loss, and it is a linear function of w. Applying the stochastic gradient descent algorithm, we have:

$$w^{(\tau+1)} = w^{(\tau)} - \eta \nabla E_P(w) = w^{(\tau)} + \eta g_n t_n \tag{7.4}$$

where η is the learning rate parameters and τ is index of the step. Without losing generality, we can set η to 1.

It is easy to interpret the perceptron algorithm. Going through the data, if it is classified correctly, weights are not changed. If it is misclassified, then for class C_1 we add vector $g(x_n)$ onto the current estimate of weights vector w, while for class C_2 we subtract. An illustration of the perceptron algorithm is shown in Figure 7.3. The color indicates class, and we have linearly separable points. After picking the initial point, and running the perceptron algorithm, we get a hyper-plane that defines two classes.
If classes are linearly separable, then the solution exists and it is found in a finite number of steps. Even if classes are linearly separable, the final solution may depend on the initialization parameters. If classes are not linearly separable then the algorithm will never converge.

The perceptron algorithm [39] does not provide probabilistic outputs, nor does it scale well for more than 2 classes. [39]

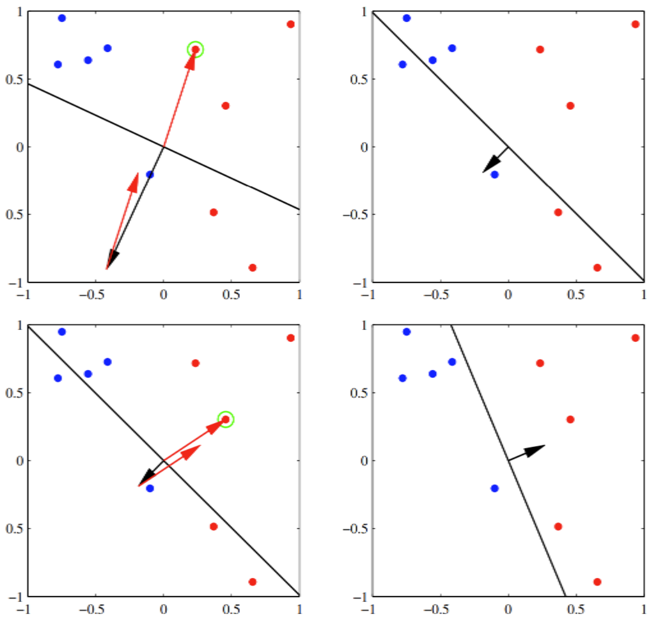


Figure 7.3: Perceptron algorithm [39]

7.1.2 Multi-layer Perceptron

Multi-layer perceptron [39] (MLP), as shown in Figure 7.4, is an expansion on the previous perceptron. In perceptron, we had only one layer, while in MLP we can extend a number of layers and thus have greater model capacity. The first and last layers are called input and output layers, respectively, while all other in between are hidden layers.

The MLP can be described as a series of functional transformations. Let $x_1, x_2, ..., x_N$ be the input data and their linear combinations:

$$a_j = \sum_{i=1}^{N} w_{ij}^{(1)} x_i + w_{j0}^{(1)} \tag{7.5}$$

where $j = 1, .., M$ and the superscript (1) indicates in which layer are parameters corresponding. In this case, it is the first layer. The parameters $w_{ji}^{(1)}$ are known are weights, and $w_{j0}^{(1)}$ are known as biases. The a_j is known as an activation. The goal is to have a nonlinear transformation function, h, which is differentiable to get:

$$z_j = h(a_j) \tag{7.6}$$

The z_j is called hidden units. For transformation functions, the most common is used *sigmoid*, *tanh*, *ReLU*, etc. Now passing the outputs from the previous to the next layer:

$$a_k = \sum_{j=1}^{M} w_{kj}^{(2)} z_j + w_{k0}^{(2)} \tag{7.7}$$

where $k = 1, ..., K$ is the total number of outputs. The final output is noted as y_k. For binary classification, the output is denoted as:

$$y_k = \sigma(a_k) \tag{7.8}$$

Putting it all together:

$$y_k = \sigma\left(\sum_{j=1}^{M} w_{kj}^{(2)} h\left(\sum_{i=1}^{N} w_{ji}^{(1)} x_i + w_{j0}^{(1)}\right) + w_{k0}^{(2)}\right) \tag{7.9}$$

For multi-class problems, a softmax function is used to get outputs.

This function can be represented in the form of a network diagram, known as feed-forward neural networks, since the result is propagated only forward, and there are no loops or cycles. [39]

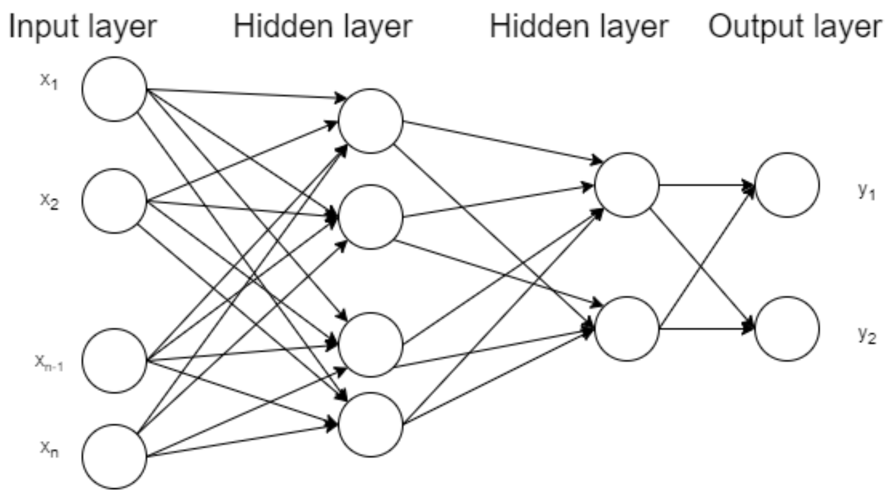


Figure 7.4: Multi-layer Perceptron

Training Network

If target vectors are $\{t_n\}$, then we minimize the error function:

$$E(w) = \frac{1}{2}\sum_{n=1}^{N} \|y(x_n, w) - t_n\|^2 \tag{7.10}$$

Let us assume that t has a Gaussian distribution with an x dependent mean:

$$p(t|x, w) = N(t|y(x, w), \beta^{-1}) \tag{7.11}$$

where β is the inverse variance of the Gaussian noise. Log-likelihood can be written as:

$$p(t|X, w, \beta) = \prod_{n=1}^{N} N(t_n|x_n, w, \beta) \tag{7.12}$$

Applying the negative logarithm, the error function is:

$$\frac{\beta}{2}\sum_{n=1}^{N}\{y(x_n, w) - t_n\}^2 - \frac{N}{2}\ln\beta + \frac{N}{2}\ln(2\pi) \tag{7.13}$$

Here it is often preferred to do minimization of an error function rather than maximization of log-likelihood. The value of w found by minimizing $E(w)$ will be w_{ML}. After substitution, the value of β can be found from:

$$\frac{1}{\beta_{\mathrm{ML}}} = \frac{1}{N}\sum_{n=1}^{N}\{y(x_n, w_{ML}) - t_n\}^2 \tag{7.14}$$

If the identity function is used as the output activation function, $y_k = a_k$, then:

$$\frac{\partial E}{\partial a_k} = y_k - t_k \tag{7.15}$$

In the case of binary classification, we have logistic sigmoid activation for the output:

$$y = \sigma(a) = \frac{1}{1 + exp(-a)} \tag{7.16}$$

Interpretation of $y(x, w)$ can be as conditional probability with $p(class_1|x)$, and $p(class_2|x) = 1 - y(x, w)$. The conditional distribution of targets can be written as:

$$p(t|x, w) = y(x, w)^t\{1 - y(x, w)\}^{1-t} \tag{7.17}$$

For a training set with independent observation, the error function is the cross-entropy loss function:

$$E(w) = -\sum_{n=1}^{N}\{t_n \ln y_n + (1 - t_n)\ln(1 - y_n)\} \tag{7.18}$$

For K labels, and K binary classification, if we assume that class labels are independent, the error function is:

$$E(w) = -\sum_{n=1}^{N}\sum_{k=1}^{K}\{t_{nk}\ln y_{nk} + (1-t_{nk})\ln(1-y_{nk})\} \tag{7.19}$$

In the case of multi-class classification, where labels are mutually exclusive and have an encoding scheme of one-hot-encoding, outputs are $y_k(x,w) = p(t_k = 1|x)$, which gives an error function:

$$E(w) = -\sum_{n=1}^{N}\sum_{k=1}^{K} t_{kn}\ln y_k(x_n, w) \tag{7.20}$$

where output is given from the softmax function:

$$y_k(x,w) = \frac{exp(a_k(x,w))}{\sum_j exp(a_j(x,w))} \tag{7.21}$$

Parameter Optimization

Let us focus now on finding the best parameters for our neural network [39]. For a small step δ in weight space from w to $w+\delta w$, then the error function is changed $\delta E \approx \delta w^T \nabla E(w)$, and vector $\nabla E(w)$ points in the direction of greatest increase in the error function. Since the error function is a continuous, smooth function of parameters w, the minimum will be at the point such that gradient error vanishes:

$$\nabla E(w) = 0 \tag{7.22}$$

If the gradient is not 0, then we make the step in the direction of $\nabla E(w)$ and reduce the error. Stationary points are the points where the gradient vanishes, and they could be minima, maxima, or saddle points, as shown in Figure 7.5. The global minimum is a point in weight space where the error function is smallest. All other minima are called local minima.

Since there are no analytical solutions for $\nabla E(w) = 0$, iterative procedures need to be applied. Usually, for initial $w^{(0)}$ are some predefined procedures. After that, weights are changing through each step:

$$w^{(\tau+1)} = w^{(\tau)} + \delta w^{(\tau)} \tag{7.23}$$

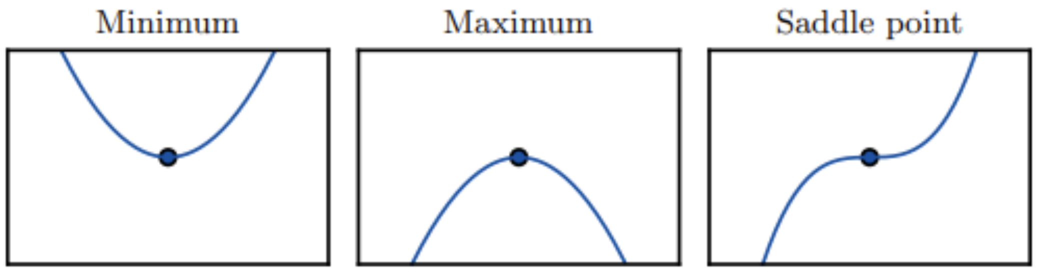


Figure 7.5: Example of minima, maxima, and saddle point [12]

If we consider Taylor expansion of $E(w)$ at some point $\hat{w}$:

$$E(w) \approx E(\hat{w}) + (w - \hat{w})^T b + \frac{1}{2}(w - \hat{w})^T H(w - \hat{w}) \tag{7.24}$$

Gradient at point $\hat{w}$ is defining b:

$$b \equiv \nabla E|_{w=\hat{w}} \tag{7.25}$$

and $H = \nabla\nabla E$ is the Hessian matrix with elements:

$$H_{ij} \equiv \frac{\partial E}{\partial w_i w_j}\bigg|_{w=\hat{w}} (7.26)$$

Now, we can approximate the local gradient as:

$$\nabla E \approx b + H(w - \hat{w}) \tag{7.27}$$

If we look at the local quadratic optimization from (7.24), around a point w^* which is minimum, then there is no linear term since $\nabla E = 0$ at w^*:

$$E(w) = E(w^*) + \frac{1}{2}(w - w^*)^T H(w - w^*) \tag{7.28}$$

Hessian is evaluated at w^*.
The gradient of the error function can be efficiently calculated by back-propagation, leading to significant improvements, as opposed to error quadratic approximation.

Using gradient information, and going in the direction of a negative gradient, weights can be updated as:

$$w^{(\tau+1)} = w^{(\tau)} - \eta\nabla E(w^{(\tau)}) \tag{7.29}$$

where parameter η is called the learning rate, and it is a positive number. After the forward pass, the gradient is calculated, weights are updated and this is repeated. If the whole dataset is used, then this is known as the batch method, and if only a subset of data is used for the pass, then this is called mini-batch gradient descent. Gradient descent or steepest descent is the direction of the greatest rate of decrease of the error function. Gradient descent in this form looks good, it's not efficient.

If the error function is represented as a sum of individual observations, then:

$$E(w) = \sum_{n=1}^{N} E_n(w) \tag{7.30}$$

Sequential gradient descent or stochastic gradient descent makes an update after each observation has been evaluated:

$$w^{(\tau+1)} = w^{(\tau)} - \eta \nabla E_n(w^{(\tau)}) \tag{7.31}$$

There is also an alternative to this, where weights are updated on a batch of points.

Back propagation

Back-propagation [39] is a term to describe the evaluation of derivatives. The first stage of back-propagation denotes evaluating derivatives and the second is updating the weights.

Let us assume we have a simple neural network with one layer, a sigmoid hidden activation function, and a sum-of-squared error. We can represent error as the sum of individual data points errors:

$$E(w) = \sum_{i=1}^{N} E_n(w) \tag{7.32}$$

If we have linear model with inputs x_i and the outputs y_i:

$$y_k = \sum_{i} w_{ki} x_i \tag{7.33}$$

and the error function:

$$E_n = \frac{1}{2}\sum_k (y_{nk} - t_{nk})^2 \tag{7.34}$$

where $y_{nk} = y_k(x_n, w)$. Now gradient with respect to the w_{ji} is:

$$\frac{\partial E_n}{\partial w_{ji}} = (y_{nj} - t_{nj})x_{ni} \tag{7.35}$$

When we run feed-forward, the output is a weighted sum of input:

$$a_j = \sum_i w_{ji} z_i \tag{7.36}$$

where z_i is the activation unit. Previous sum in (7.36) is transformed with nonlinear activation function h to get activation z_j:

$$z_j = h(a_j) \tag{7.37}$$

This process is called forward propagation [39].

Now, let us consider derivatives of the E_n with respect to the weight w_{ij}. Subscript n will be omitted so it is more clearly written. Error E_n depends on the weight w_{ij} only on summed input a_j to unit j. Applying the chain rule for partial derivatives we get:

$$\frac{\partial E_n}{\partial w_{ji}} = \frac{\partial E_n}{\partial a_j}\frac{\partial a_j}{\partial w_{ji}} \tag{7.38}$$

If we set:

$$\delta_j \equiv \frac{\partial E_n}{\partial a_j} \tag{7.39}$$

δ is referred to as an error. Using using (7.37), we can write:

$$\frac{\partial a_j}{\partial w_{ji}} = z_i \tag{7.40}$$

And finally, using (7.38), (7.39), and (7.40):

$$\frac{\partial E_n}{\partial w_{ji}} = \delta_j z_i \tag{7.41}$$

The previous equation (7.41) tells us that in order to get the derivative, we need to multiply the value δ with the output end of the weight. In order to evaluate the derivatives, we need only to calculate the value of δ_j for each hidden and output unit in the network, and then apply the equation.

For the final layer:

$$\delta_k = y_k - t_k \tag{7.42}$$

For the hidden layers:

$$\delta_j \equiv \frac{\partial E_n}{\partial a_j} = \sum_k \frac{\partial E_n}{\partial a_k} \frac{\partial a_k}{\partial a_j} \tag{7.43}$$

where sum is evaluated over all *units* in which j sends connection. Substituting δ from (7.39) in (7.40), and use with previous equations (7.36) and (7.37):

$$\delta_j = h'(a_j) \sum_k w_{kj} \delta_k \tag{7.44}$$

Algorithm 3 Back-propagation algorithm [39]

```
1: For input vector $x_n$ run forward pass
2: Evaluate $\delta_k$ for all output units
3: Back-propagate the $\delta'$s to obtain $\delta_j$ for each hidden unit
4: Evaluate derivatives using (7.41)
```

7.2 RNN

MLP suffers from a couple of things. As the input size is increased, the number of hidden layers and their size, and computational complexity grows exponentially, thus making them hard to train. On top of that, they do not utilize the possibility of dependencies between input features. In text modeling, there is dependence between consecutive input words and that is what recurrent neural networks try to solve.

Recurrent neural networks [12] (RNNs), are a family of neural networks for processing sequential data. Let RNNs operate on a sequence $x^{(t)}$ with the time step index t in the range from 1 to τ. Now the computational graph is extended to cycles since the input of a cell can be the output of the same cell, in the previous moment.

Let the dynamical system be:

$$s^{(t)} = f(s^{(t-1)}; \theta) \tag{7.45}$$

where $s^{(t)}$ is called the state of the system at moment t, and it is recurrent because time t refers back to the same definition at time $t-1$.

For example, if set $t = 3$, and unfold sequence, we get:

$$s^{(3)} = f(s^{(2)}; \theta) = f(f(s^{(1)}; \theta); \theta) \tag{7.46}$$

Let us know to consider a dynamical system that is driven by external signal x:

$$s^{(t)} = f(s^{(t-1)}, x^{(t)}; \theta) \tag{7.47}$$

In order to make an indication that the state is the hidden unit of the network, we can use h for it, and rewrite the previous equation:

$$h^{(t)} = f(h^{(t-1)}, x^{(t)}; \theta) \tag{7.48}$$

Hidden states try to capture information from the input, and this is generally lossy because maps arbitrary sequence length to a fixed vector size.

If we change a bit representation at time step t with function $g^{(t)}$:

$$h^{(t)} = g^{(t)}(x^{(t)}, ..., x^{(1)}) = f(h^{(t-1)}, x^{(t)}; \theta) \tag{7.49}$$

Now, the function $g^{(t)}$ takes all the previous sequence $(x^{(t)}, ..., x^{(1)})$ as input and gives output current state. No matter what the sequence length is, the model has the same input size, and it is possible to use the same transition function f at each state.

7.2.1 Vanilla RNN

There are a couple of different variations of recurrent neural networks [12]:

1. Recurrent neural networks that produce an output at each step and have the recurrent connection between hidden layers, which is shown in Figure 7.6.
2. Recurrent neural networks that produce an output at each step and have recurrent connection only from the output of one-time step to the hidden in the next time stamp, which is shown in Figure 7.7.
3. Recurrent neural networks with recurrent connections between hidden units, read entire sequences and produce only one output, which is shown in Figure 7.8.

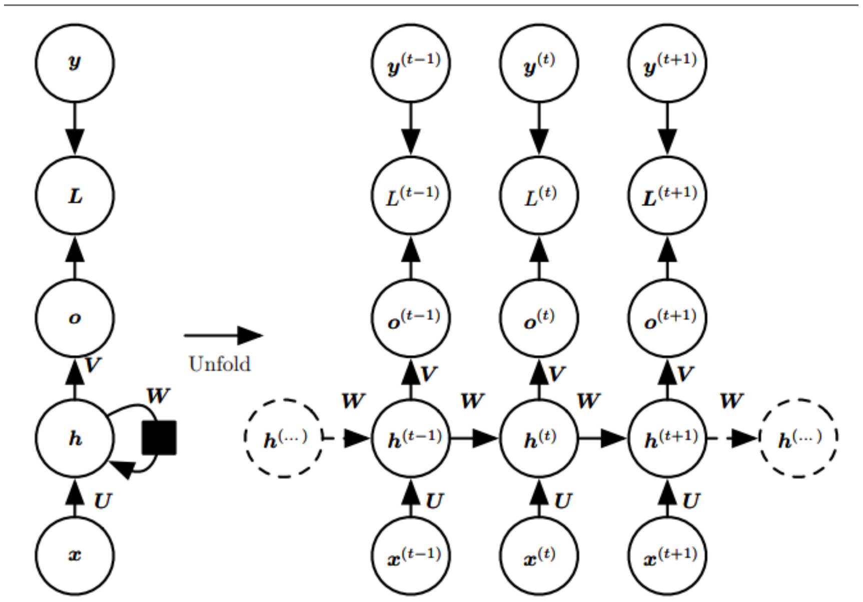


Figure 7.6: RNN type 1 [12]

Let us assume that the activation function is a hyperbolic tangent. Also, let us assume that the output is discrete (i.e. words). Output o can be represented as an unnormalized log probability of each discrete value. After that, the softmax function can be applied to get vector $\hat{y}$ of normalized probabilities. Let $h^{(0)}$ be the initial state, then for time stamps from $t = 1$ to $t = \tau$ we get next equations:

$$a^{(t)} = b + Wh^{(t-1)} + Ux^{(t)} \tag{7.50}$$

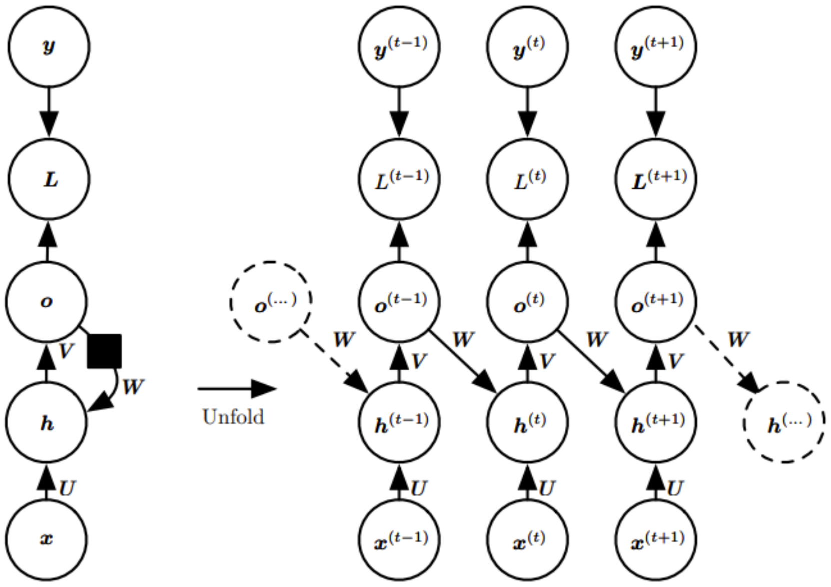

y
L
o
V
W
h
U
x
Unfold
$y^{(t-1)}$ $y^{(t)}$ $y^{(t+1)}$
$L^{(t-1)}$ $L^{(t)}$ $L^{(t+1)}$
$o^{(\dots)}$ $o^{(t-1)}$ $o^{(t)}$ $o^{(t+1)}$
$h^{(t-1)}$ $h^{(t)}$ $h^{(t+1)}$ $h^{(\dots)}$
$x^{(t-1)}$ $x^{(t)}$ $x^{(t+1)}$
W V U

Figure 7.7: RNN type 2 [12]

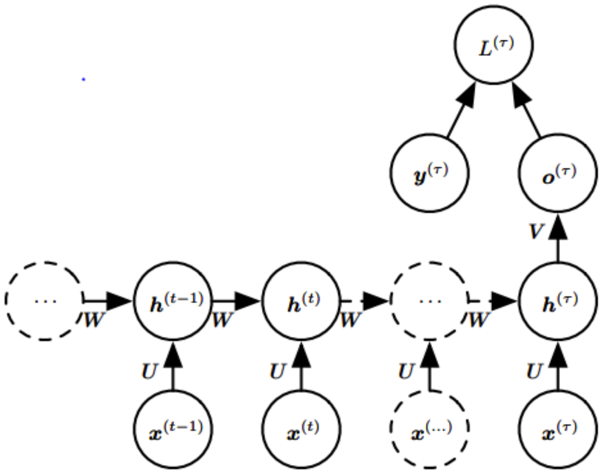

$L^{(\tau)}$
$y^{(\tau)}$ $o^{(\tau)}$
V
… $h^{(t-1)}$ $h^{(t)}$ … $h^{(\tau)}$
W U
$x^{(t-1)}$ $x^{(t)}$ $x^{(\dots)}$ $x^{(\tau)}$

Figure 7.8: RNN type 3 [12]

$$h^{(t)} = \tanh(a^{(t)}) \tag{7.51}$$

$$o^{(t)} = c + Vh^{(t)} \tag{7.52}$$

$$\hat{y}^{(t)} = \text{softmax}(o^{(t)}) \tag{7.53}$$

where b and c are bias vectors and matrices U, V and W weights for input-hidden, hidden-output, and hidden-hidden sequences. If $L^{(t)}$ is loss, then the negative log-likelihood of $y^{(t)}$ given $x^{(1)}$:

$$\begin{aligned} L(\{x^{(1)}, ..., x^{(\tau)}\}, \{y^{(1)}, ..., y^{(\tau)}\}) = \\ &= \sum_t L^{(t)} \\ &= -\sum_t \log p_{model}(y^{(t)} | \{x^{(1)}, ..., x^{(\tau)}\}) \end{aligned} \tag{7.54}$$

where $\log p_{\text{model}}(y^{(t)} | \{x^{(1)}, ..., x^{(\tau)}\})$ is the entry from the output vector $\hat{y}^{(t)}$.

The gradient of this loss with respect to the parameters is a very expensive operation because forward bass needs to be made, followed by a backward propagation pass, and this is a sequential operation. The back-propagation applied to the unrolled graph is called back-propagation through time (BPTT) [12].

Back-propagation through time (BPTT)

We assume that outputs $o^{(t)}$ are input to the softmax function to get the vector of probabilities $\hat{y}$. Loss is the negative log-likelihood of real target values $y^{(t)}$, for a given input. Applying chain rule:

$$\frac{\partial L}{\partial V} = \sum_{t=1}^{\tau} \frac{\partial L^{(t)}}{\partial V} = \sum_{t=1}^{\tau} \frac{\partial L^{(t)}}{\partial o_t} \frac{\partial o^{(t)}}{\partial V} \tag{7.55}$$

If case when we want to find the gradient with respect to W at timestamp t for hidden state $h^{(t)}$, it is not straightforward as in the previous case, since state $h^{(t)}$ also depends on the previous state $h^{(t-1)}$.

$$\frac{\partial L^{(t)}}{\partial W} = \frac{\partial L^{(t)}}{\partial o^{(t)}} \frac{\partial o^{(t)}}{\partial h^{(t)}} \frac{\partial h^{(t)}}{\partial h^{(t-1)}} \frac{\partial h^{(t-1)}}{\partial W} \tag{7.56}$$

$$\frac{\partial L}{\partial W} = \sum_{t=1}^{\tau} \frac{\partial L^{(t)}}{\partial o^{(t)}} \frac{\partial o^{(t)}}{\partial h^{(t)}} \frac{\partial h^{(t)}}{\partial h^{(t-1)}} \frac{\partial h^{(t-1)}}{\partial W} \tag{7.57}$$

Gradients for the input weight, in the same way, we get:

$$\frac{\partial L}{\partial V} = \sum_{t=1}^{\tau} \frac{\partial L^{(t)}}{\partial o^{(t)}} \frac{\partial o^{(t)}}{\partial h^{(t)}} \frac{\partial h^{(t)}}{\partial h^{(t-1)}} \frac{\partial h^{(t-1)}}{\partial V} \tag{7.58}$$

7.2.2 Bidirectional Networks

In all previous approaches, we have only discussed that at time step (t) we pass collected information at $(t-1)$. Sometimes, for a better prediction, we need to know the whole input sequence to make a better judgment (this is particularly important for speech-to-text).
Bidirectional RNNs [12], as in Figure 7.9, are using a combination of forward and backward input sequences through time.

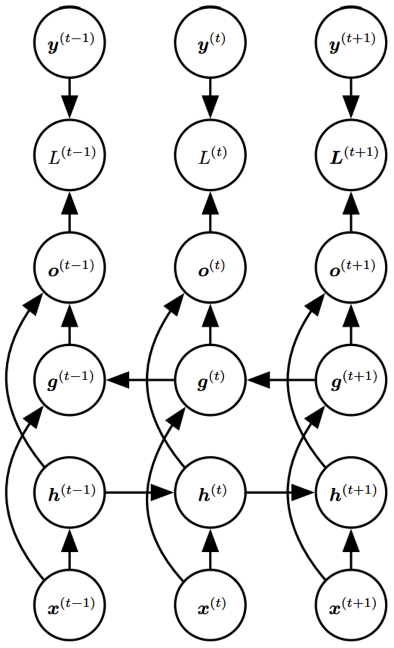


Figure 7.9: Bidirectional Recurrent Network [12]

7.2.3 LSTM

Of all RNN variations, the most used are gated RNN, which includes long short-term memory [12], amongst others (gated recurrent networks). They are based on creating paths through time so the gradients do not explode or vanish. They allow networks to accumulate information.

The input gate determines how much of the input node's value should be added to the current memory cell's internal state. The forget gate determines whether to keep the current value of the memory or flush it. And the output gate determines whether the memory cell should influence the output at the current time step. [41]

Let the state cell be $s_i^{(t)}$, which is controlled by forgetting gate unit $f_i^{(t)}$, where t is time step, $x^{(t)}$ is the current input vector and $h^{(t)}$ is currently hidden layer vector, containing the outputs of all LSTM cells, and b^f, U^f, and W^f are respectively biases, input weights and recurrent wights for the forget gate. The external input gate is noted as $g_i^{(t)}$, and $q_i^{(t)}$ is output gate. LSTM cell is now updated by:

$$s_i^{(t)} = f_i^{(t)} s_i^{(t-1)} + g_i^{(t)} \sigma(b_i + \sum_j U_{i,j} x_j^{(t)} + \sum_j W_{i,j} h_j^{(t-1)}) \tag{7.59}$$

$$f_i^{(t)} = \sigma(b_i^f + \sum_j U_{i,j}^f x_j^{(t)} + \sum_j W_{i,j}^f h_j^{(t-1)}) \tag{7.60}$$

$$g_i^{(t)} = \sigma(b_i^g + \sum_j U_{i,j}^g x_j^{(t)} + \sum_j W_{i,j}^g h_j^{(t-1)}) \tag{7.61}$$

$$h_i^{(t)} = \tanh(s_i^{(t)}) q_i^{(t)} \tag{7.62}$$

$$q_i^{(t)} = \sigma(b_i^o + \sum_j U_{i,j}^o x_j^{(t)} + \sum_j W_{i,j}^o h_j^{(t-1)}) \tag{7.63}$$

An example of an LSTM cell is given in Figure 7.10.

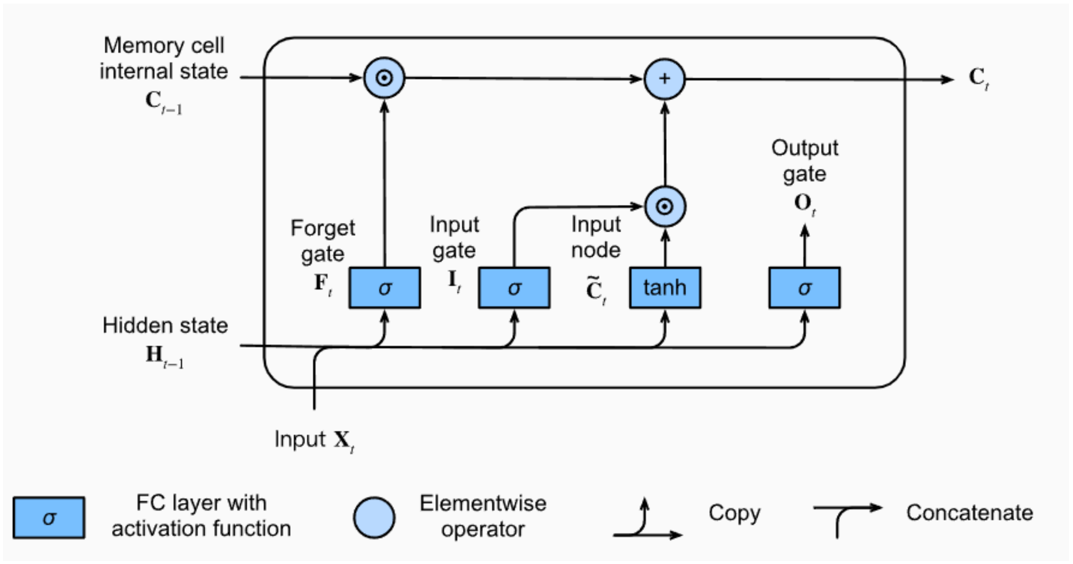


Figure 7.10: LSTM Cell [41]

7.3 Transformers

7.3.1 Attention Mechanism

RNN, LSTM, and GRU have been established ad state of the art techniques in sequence modeling. Recurrent models use their output as input, therefore making it sequentially and without the possibility for parallelization. Attention mechanisms [14] are developed for sequence modeling, allowing dependencies without regard to their distance in the input or output.

7.3.2 Model Architecture

Let the input sequence be $(x_1, ..., x_n)$, and encoded sequence $z = (z_1, ..., z_n)$. Given z, the decoder then generates an output sequence $(y_1, ..., y_m)$ of symbols one element at a time. The model in Figure 7.11 is using previous output also as an input. [14]

Encoder: The encoder is composed of $N = 6$ identical layers. Each layer has two sub-layers. The first is a multi-head self-attention mechanism, and the second is a simple, position-wise fully connected feed-forward network. There are residual connections between two sub-layers, followed by a normalization layer.

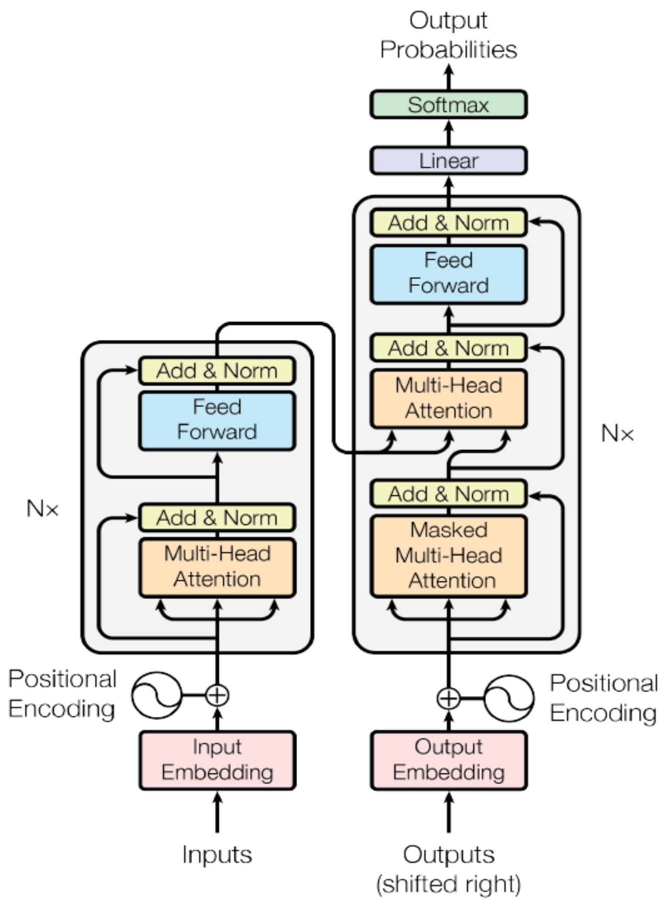


Figure 7.11: Attention model architecture [14]

Decoder: The decoder is similar to the encoder with $N = 6$ identical layers. In the sub-layer of the decoder, on top of all sub-layers in the encoder, the decoder also has a third sub-layer that performs.

An attention function can be described as mapping a query and a set of key-value pairs to an output, where the query, keys, values, and output are all vectors. The output is computed as a weighted sum of the values, where the weight assigned to each value is computed by a compatibility function of the query with the corresponding key.

Attention in Figure 7.12 consists of keys and queries are dimensions d_k and values are dimensions d_v. The dot product is computed of queries and keys and normalized with $\sqrt{d_k}$. After division, values are run through the softmax function. Let K, V, and Q be the matrices of keys, values, and queries. Then, the output is calculated as follows:

$$\text{Attention(Q, K, V)} = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V \tag{7.64}$$

Instead of doing single attention, multi-head attention, as in Figure 7.13, allows different representations to be captured jointly.

$$\text{MultiHead(Q, K, V)} = \text{Concat}(head_1, ..., head_h)W^O \tag{7.65}$$

where

$$\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \tag{7.66}$$

Positional encoding is defined as follows:

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{\frac{2i}{d_{model}}}}\right)$$
$$PE_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{\frac{2i}{d_{model}}}}\right)$$

There are a couple of reasons to use the attention mechanism compared to RNN: total complexity per layer, amount of parallelization, and ability to capture long-range dependencies. Also, it is more interpretable, since we can visualize which words are more contributing to overall prediction.

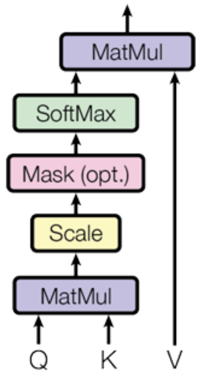


Figure 7.12: Attention model block [14]

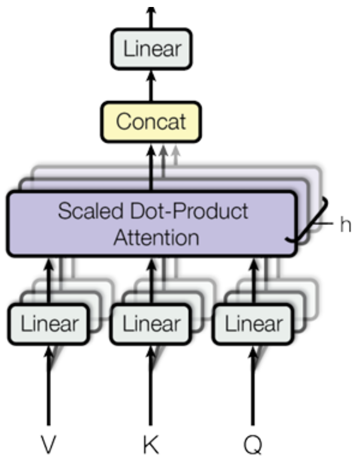


Figure 7.13: Multiple attention model block [14]

7.3.3 BERT

BERT [15] (Bidirectional Encoder Representations from Transformers) is a framework built on top of the attention mechanism. It consists of two phases: *pre-training* and *fine-tuning*. In the pre-training phase, the model is trained on unlabeled data, on a couple of pre-training tasks. In the fine-tuning phase, the initial model is taken and trained with downstream tasks.

Model architecture

BERT's model architecture is a multi-layer bidirectional Transformer encoder based on the attention mechanisms. Let the number of layers be L, the hidden size as H, and the number of self-attention heads as A. Then, $\mathbf{BERT}_{BASE}$ consists from L=12, H=768, A=12, and total number of parameters=110M.

In order to make BERT work in any downstream tasks, the model is able to handle both a single sentence and a pair of sentences. The first token of every sequence is a special token noted with $[CLS]$. Sentence pairs are packed together with separation token $[SEP]$. Also, all tokens in one sequence have an indication of which sequence tokens belong. If we note E as the input embedding, special token $[CLS]$ token as $C \in \mathbb{R}^H$ and the $T_i \in \mathbb{R}^H$ input token.

BERT pre-training tasks

Task 1. Masked LM To train a deep bidirectional representation, we simply mask some percentage of the input tokens at random and then predict those masked tokens. We refer to this procedure as a "masked LM". Around 15% of the tokens are masked randomly. This comes with a little cost since in the fine-tuning, there are no $[MASK]$ tokens. Instead, if i-th token is selected for masking, there is an 80% chance to mask it with a mask token, a 10% chance to change it with the random token, and a 10% chance to stay the same.

Task 2. Next Sentence Prediction (NSP) The model is trained to learn the next sentence prediction task with binary outputs. During the training process, 50% of sentences are the true next sentences, and 50% of sentences are random sentences.

Fine-tuning BERT

Fine-tuning BERT is a simple procedure. For each task, the inputs and outputs are passed into BERT and all parameters are fine-tuned.
Compared to pre-training, fine-tuning is relatively inexpensive.

Pre-training data used for BERT are BooksCorpus (800M words) and English Wikipedia (2,500M words) [15]. From pre-trained BERT, the embedding matrix is extracted and used for vectorization with the classification model.

Chapter 8

Optimization

Optimization of neural networks is focused on one thing: finding weight parameters in order to reduce loss [12]. Optimization is done using gradient-based principles.

8.1 Stochastic Gradient Descent (SGD)

Let $L(w)$ be the objective function of parameters of the model, and η be the learning rate. Gradient descent is an optimization technique [42] to minimize an objective function $L(w)$ by updating the parameters in the opposite direction of the gradient of the objective function $\nabla L(w)$ w.r.t. to the parameters. Stochastic gradient descent (SGD) performs a parameter update for each mini-batch of m training example.

Algorithm 4 Stochastic gradient descent [12]

```
Require: Learning rate $\eta$
Require: Initial parameter $w$
  for epoch in range of maximum epochs do do
      Sample a mini-batch of m examples from the training set $\{x^1, ..., x^m\}$
  with corresponding targets $y^i$ .
      Compute gradient estimate $g \leftarrow \frac{1}{m}\nabla_w \sum_i L(f(x^i; w), y^i)$.
      Apply update: $w \leftarrow w - \eta g$
  end for
```

To guarantee the convergence of SGD [12], the next conditions need to be satisfied:

$$\sum_{k=1}^{\infty} \eta = \infty \tag{8.1}$$

$$\sum_{k=1}^{\infty} \eta^2 < \infty \tag{8.2}$$

Deciding what would be the learning rate is usually done after a couple of iterations of experiments, and it is desired to decrease it over time. If it is too low, the model will not converge. If it is too high, too many oscillations will be done.

In Figure 8.1 blue lines represent points with the same loss, and minima is in the middle. The initial point is x_0. SGD calculates the gradient and updates the weights in the opposite direction (red lines). Repeating this procedure we get the sequence of points (x_1, x_2, x_3, x_4), going to the minima.

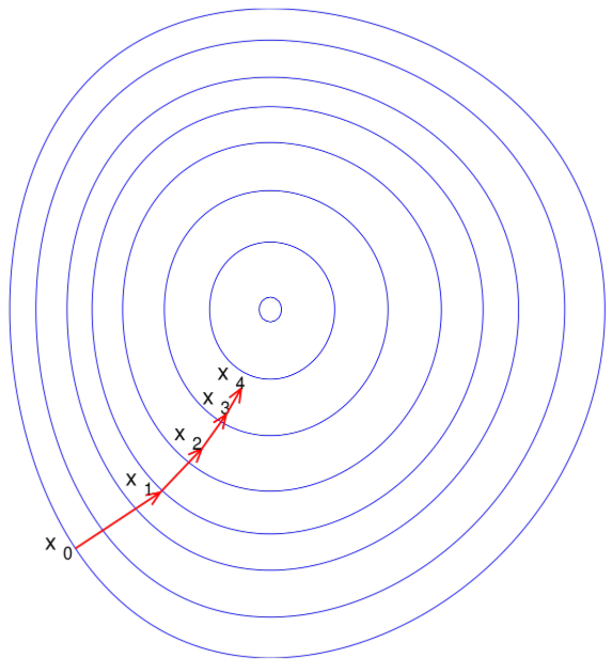


Figure 8.1: Gradient descent
[43]

8.2 Momentum

SGD does not capture information about past gradients that could speed up the optimization part, especially in the face or high curvature. The momentum algorithm [12] accumulates an exponentially decaying moving average of past gradients and continues to move in their direction. Formally, the momentum algorithm introduces a variable v that plays the role of velocity—it is the direction and speed at which the parameters move through parameter space. The velocity is set to an exponentially decaying average of the negative gradient.

In Figure 8.2, we have a quadratic loss and examples of the convergence of SGD with and without momentum. The plain gradient goes back and forth, makes large jumps and it takes more time to converge.

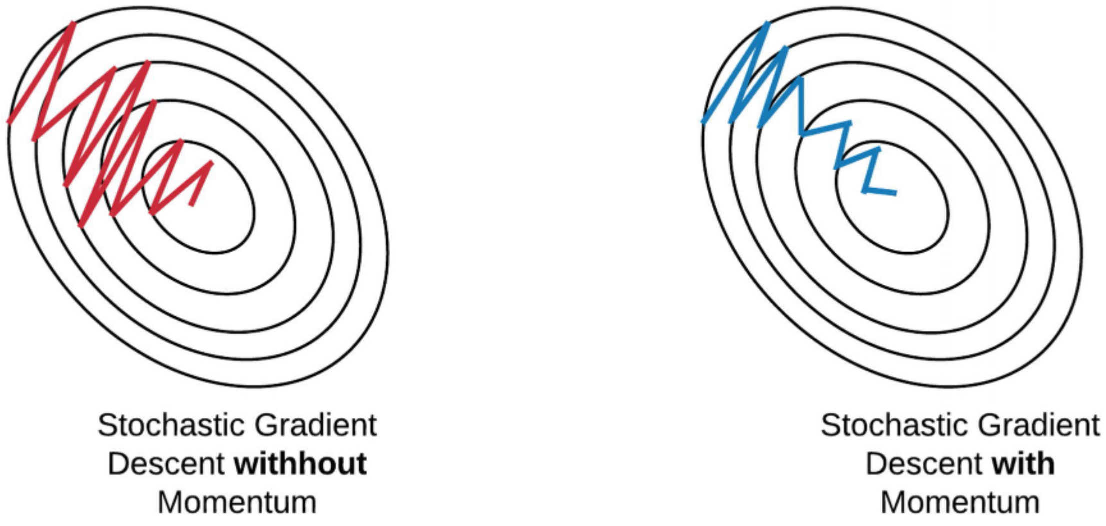

Stochastic Gradient Descent **withhout** Momentum
Stochastic Gradient Descent **with** Momentum

Figure 8.2: SGD without and with momentum
[44]

8.3 Algorithms with Adaptive Learning Rates

8.3.1 AdaGrad

The AdaGrad adapts the learning rate to the o the parameters, performing larger updates for infrequent and smaller updates for frequent parameters.

Algorithm 5 Stochastic gradient descent with momentum [12]

```
Require: Learning rate $\eta$, momentum parameter $\alpha$
Require: Initial parameter $w$, initial velocity $v$
  for epoch in range of maximum epochs do do
      Sample a mini-batch of m examples from the training set $\{x^1, ..., x^m\}$
  with corresponding targets $y^i$ .
      Compute gradient estimate $g \leftarrow +\frac{1}{m}\nabla_w \sum_i L(f(x^i; w), y^i)$
      Compute velocity update $v \leftarrow \alpha v - \epsilon g$
      Apply update: $w \leftarrow w + v$
  end for
```

It is well-suited for dealing with sparse data [42].

Algorithm 6 The AdaGrad Algorithm [12]

```
Require: Global learning rate $\eta$
Require: Initial parameter $w$
Require: Small constant $\delta$ for numerical stability
  Initialize gradient accumulation variable $r = 0$
  for epoch in range of maximum epochs do do
      Sample a mini-batch of m examples from the training set $\{x^1, ..., x^m\}$
  with corresponding targets $y^i$.
      Compute gradient estimate $\leftarrow \frac{1}{m}\nabla_w \sum_i L(f(x^i; w), y^i)$
      Accumulate squared gradient: $r \leftarrow r + g \odot g$
      Compute update: $\nabla w \leftarrow -\frac{\eta}{\sqrt{\delta + r}} \odot g$ (Division and square root are
  applied element-wise)
      Apply update: $w \leftarrow w + \nabla w$
  end for
```

8.3.2 RMSProp

The RMSProp algorithm [12] is a modification of the AdaGrad algorithm, to perform better in nonconvex problems. Instead of accumulating gradients, it takes an exponentially weighted moving average.

Algorithm 7 The RMSProp Algorithm [12]

```
Require: Global learning rate $\eta$, decay rate $\rho$
Require: Initial parameter $w$
Require: Small constant $\delta$ for numerical stability
  Initialize gradient accumulation variable $r = 0$
  for epoch in range of maximum epochs do do
      Sample a mini-batch of m examples from the training set $\{x^1, ..., x^m\}$
  with corresponding targets $y^i$.
      Compute gradient estimate $g \leftarrow +\frac{1}{m}\nabla_w \sum_i L(f(x^i; w), y^i)$
      Accumulate squared gradient: $r \leftarrow \rho r + (1 - \rho) g \odot g$
      Compute update: $\nabla w \leftarrow -\frac{\eta}{\sqrt{\delta + r}} \odot g$
      Apply update: $w \leftarrow w + \nabla w$
  end for
```

Chapter 9

Experiments and Results

9.1 Hardware

All the experiments use Python programming language, with Anaconda and PyCharm as the environments. The graphics processing unit used was NVIDIA GeForce GTX 1080 Ti with 3584 CUDA cores with 11GB memory, AMD Ryzen 3700x central processing unit with 8 cores and 16 threads, and 32GB of random access memory.

9.2 Libraries

For the preprocessing, the following libraries are used: NumPy, SpaCy, NLTK, Tweet preprocessor, Gensim, HuggingFace, and Transformers. For the training, the following libraries are used: Scikit-Learn and PyTorch. For visualization, the following libraries are used: Tensorboard, Matplotlib, and Seaborn.

9.3 Results and Discussion

All models are trained using 10-fold validation and results are summed across all validation folds. Metrics used for comparisons are accuracy, recall, precision, and F1. Machine learning models from scikit-learn are found using a random grid search for the best results. In the case of deep learning models, only one set of parameters is used for training all models on all datasets, since time and resources have been limited.

Text is first normalized using preprocessing techniques and saved in two formats: cleaned text and tokenized cleaned text, since BERT embedding requires input to be string, while other embeddings require input to be tokenized. The following embedding techniques are used for text vectorization:

- TF-IDF
- Word2Vec
- GloVe
- FastText
- Custom embeddings
- BERT embeddings

Vectorized text is used as input to the following classification models:

- Naive Bayes
- Logistic regression
- SVM
- Random Forest
- LSTM

Embedding vocabulary and dimensions:

- TF-IDF: 18000 words (each word had to appear at least 10 times) and the same embedded dimension
- Word2Vec: 300000 words and 300 embedded dimension
- GloVe: 40001 words and 300 embedded dimension
- FastText: 100000 words and 300 embedded dimension
- Custom embeddings: 30001 words and 300 embedded dimension
- BERT embeddings: 30572 tokens and 768 embedded dimension

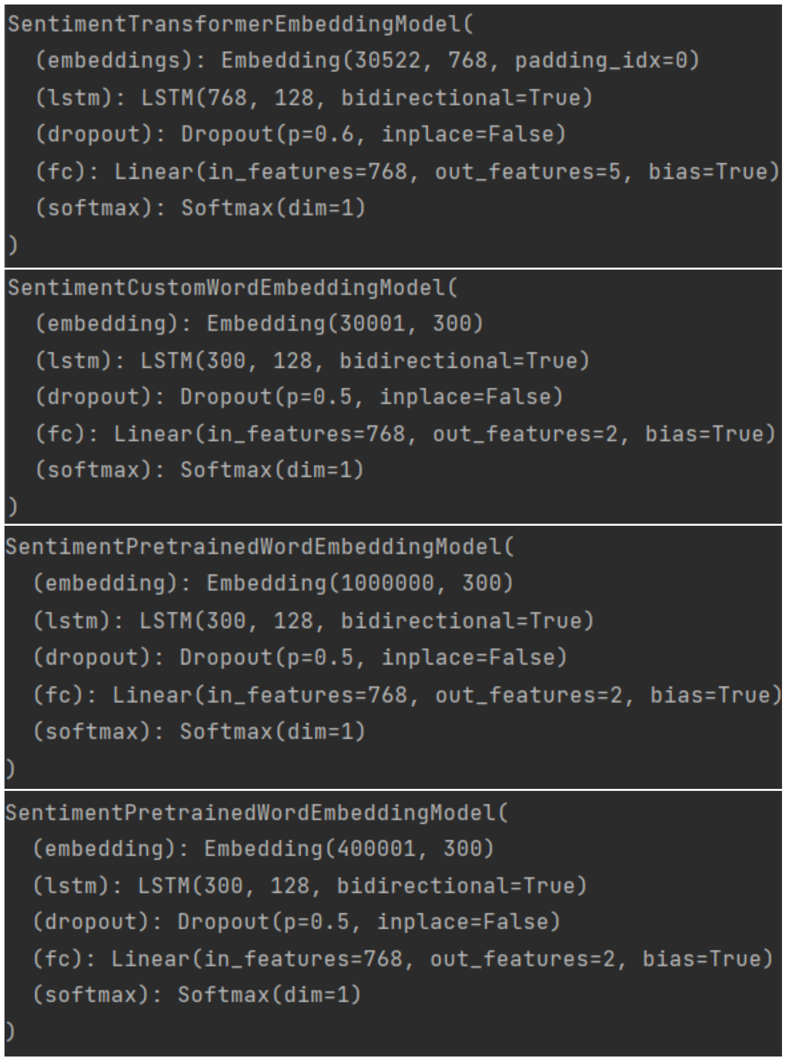
LSTM model is used with all embeddings, except word2vec because of the size, while other models are used with all embeddings except custom and BERT embeddings.

In the case of pretranied embedding, words that do not appear in the vocabulary (OOV words) are handled in such a way, that a random vector is initialized and assigned to all of them. In comparison, BERT splits each word into subpart and then create word embedding. For pretrained word embeddings and BERT embeddings, the embedding layer was frozen during the training, but it can also be fine-tuned for better embeddings, for a specific use case.

LSTM hyper-parameters used for the training:

- Hidden dimension: 128
- Last ANN layer dropout: 0.5
- Bidirectional: True
- Learning rate: 0.0006
- Batch size: 64
- Number of epochs: 30
- Maximum sequence length: 512

For hidden dimension 128 and bidirectional LSTM, LSTM output is a matrix of shape maximum sequence length and 256, where vectors from both directions are concatenated. These outputs are combined in multiple ways to get a richer representation. Three different vectors are concatenated: last LSTM output averaged vectors and mean vectors, similar to pooling operations in convolutional neural networks.
All deep neural networks model summaries are shown in Figure 9.1.



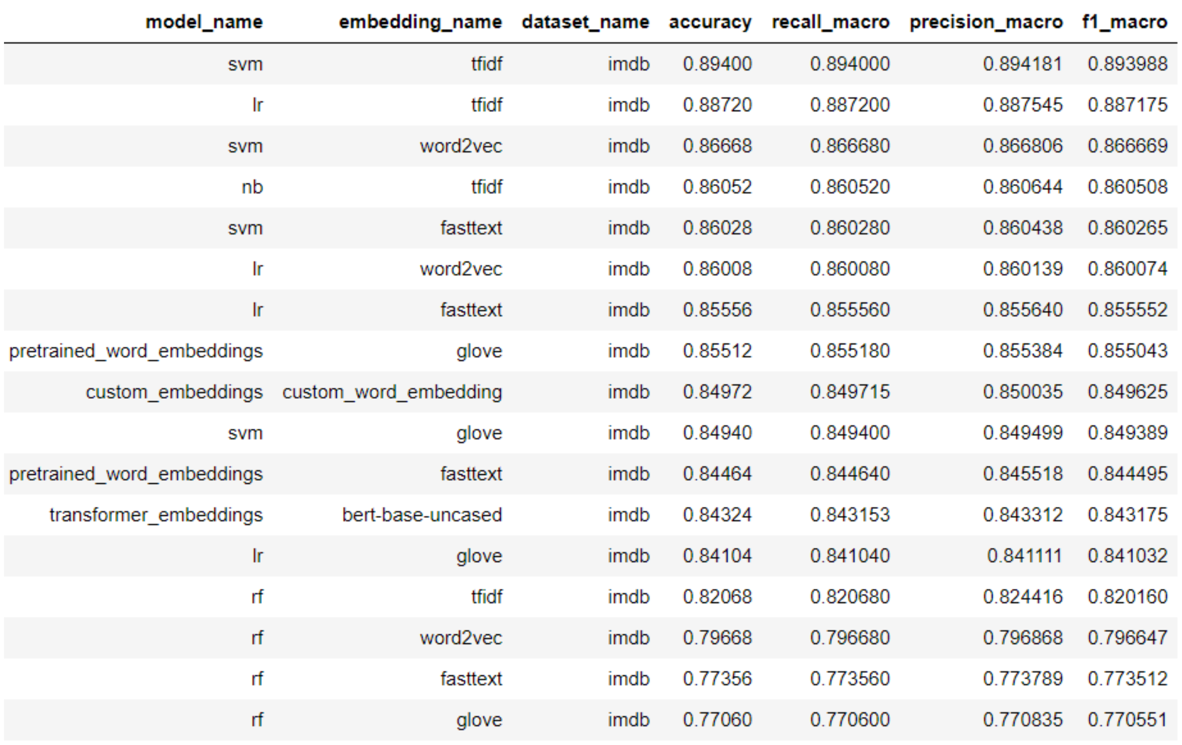
```
SentimentTransformerEmbeddingModel(
  (embeddings): Embedding(30522, 768, padding_idx=0)
  (lstm): LSTM(768, 128, bidirectional=True)
  (dropout): Dropout(p=0.6, inplace=False)
  (fc): Linear(in_features=768, out_features=5, bias=True)
  (softmax): Softmax(dim=1)
)
SentimentCustomWordEmbeddingModel(
  (embedding): Embedding(30001, 300)
  (lstm): LSTM(300, 128, bidirectional=True)
  (dropout): Dropout(p=0.5, inplace=False)
  (fc): Linear(in_features=768, out_features=2, bias=True)
  (softmax): Softmax(dim=1)
)
SentimentPretrainedWordEmbeddingModel(
  (embedding): Embedding(1000000, 300)
  (lstm): LSTM(300, 128, bidirectional=True)
  (dropout): Dropout(p=0.5, inplace=False)
  (fc): Linear(in_features=768, out_features=2, bias=True)
  (softmax): Softmax(dim=1)
)
SentimentPretrainedWordEmbeddingModel(
  (embedding): Embedding(400001, 300)
  (lstm): LSTM(300, 128, bidirectional=True)
  (dropout): Dropout(p=0.5, inplace=False)
  (fc): Linear(in_features=768, out_features=2, bias=True)
  (softmax): Softmax(dim=1)
)
```

Figure 9.1: (a) Custom embeddings model (b) Pretrained FastText model (c) Pretranied GloVe model (d) BERT embeddings model

9.4 Experiment results

9.4.1 IMDB reviews results

The results for the IMDB reviews dataset are displayed in Figure 9.2. The maximum achieved metric is 89.4% F1-macro score. A notable mention is the LR classifier with TF-IDF, with 88.7% F1-score. After that, we have a drop in the F1 score. The lowest accuracy is achieved with the RF classifier and GloVe embeddings, with a 77% F1-score. There is around a 12% difference between the best and the worst model.



model_name	embedding_name	dataset_name	accuracy	recall_macro	precision_macro	f1_macro
svm	tfidf	imdb	0.89400	0.894000	0.894181	0.893988
lr	tfidf	imdb	0.88720	0.887200	0.887545	0.887175
svm	word2vec	imdb	0.86668	0.866680	0.866806	0.866669
nb	tfidf	imdb	0.86052	0.860520	0.860644	0.860508
svm	fasttext	imdb	0.86028	0.860280	0.860438	0.860265
lr	word2vec	imdb	0.86008	0.860080	0.860139	0.860074
lr	fasttext	imdb	0.85556	0.855560	0.855640	0.855552
pretrained_word_embeddings	glove	imdb	0.85512	0.855180	0.855384	0.855043
custom_embeddings	custom_word_embedding	imdb	0.84972	0.849715	0.850035	0.849625
svm	glove	imdb	0.84940	0.849400	0.849499	0.849389
pretrained_word_embeddings	fasttext	imdb	0.84464	0.844640	0.845518	0.844495
transformer_embeddings	bert-base-uncased	imdb	0.84324	0.843153	0.843312	0.843175
lr	glove	imdb	0.84104	0.841040	0.841111	0.841032
rf	tfidf	imdb	0.82068	0.820680	0.824416	0.820160
rf	word2vec	imdb	0.79668	0.796680	0.796868	0.796647
rf	fasttext	imdb	0.77356	0.773560	0.773789	0.773512
rf	glove	imdb	0.77060	0.770600	0.770835	0.770551

Figure 9.2: IMDB models results

The best results are achieved with the TF-IDF vectorization technique with the parameters:

- max_features: 30000
- min_df: 10

- stop_words: english

and classification model SVM with the following parameters:

- C: 10
- kernel: rbf

Looking at the F1 score, the best model performance is very good, even though there are only two classes in the dataset. From the confusion matrix, we can conclude that both classes are captured with similar accuracy, with 90% for negative reviews and 89% for positive reviews. The confusion matrix with raw number is shown in Figure 9.3, and the normalized version in Figure 9.4.

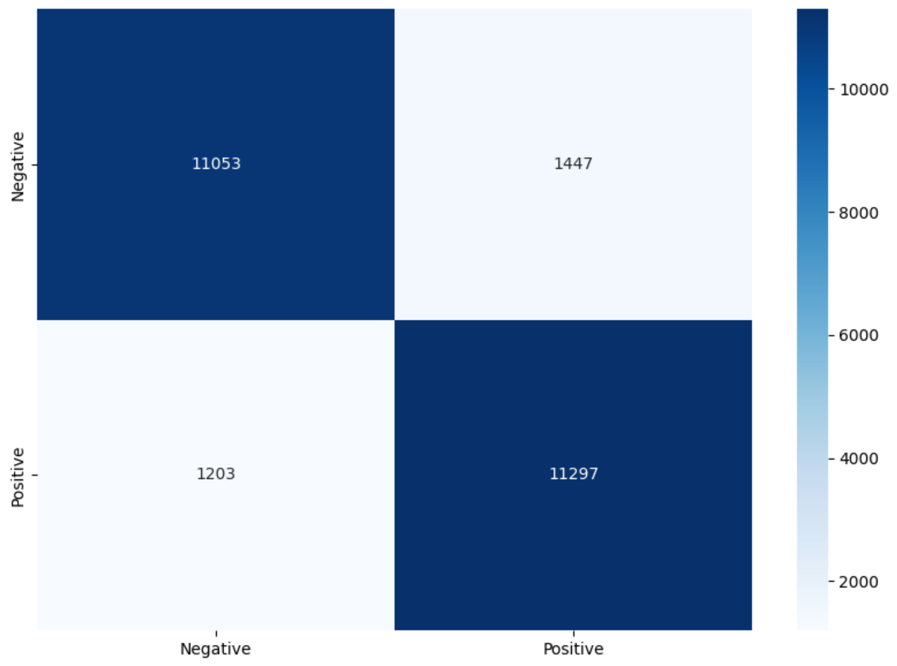


Figure 9.3: IMDB confusion matrix

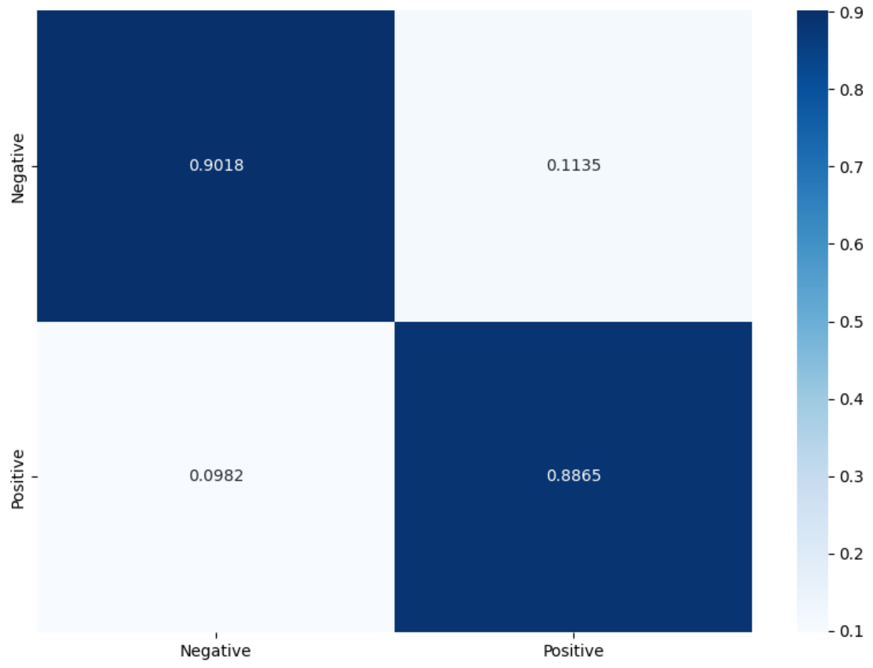


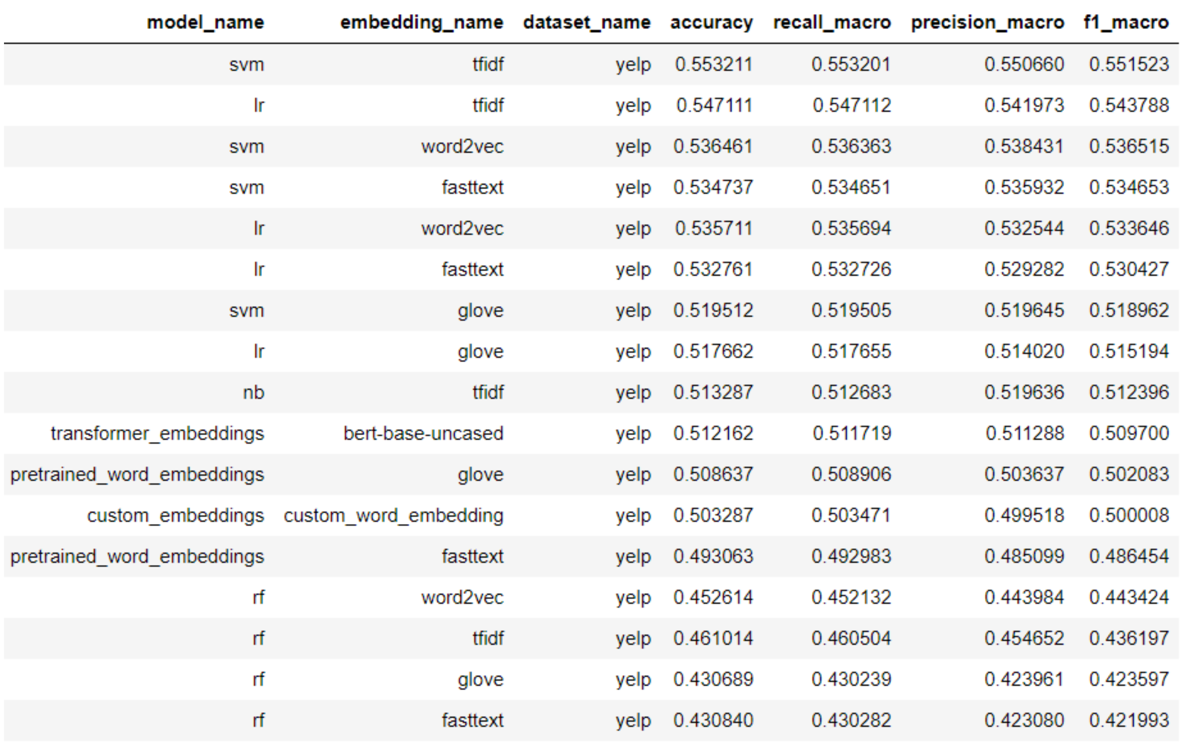
Figure 9.4: IMDB normalized confusion matrix

9.4.2 Yelp reviews results

The results for the YELP reviews dataset are displayed in Figure 9.5.
The maximum achieved metric is a 55.15% F1-macro score. Notable mentions mention is the LR classifier with TF-IDF vectorization with 54.3% F1-score. The lowest achieved metric score is 42.1% with the RF classifier and FastText. There is a 12% difference between the best and worst models.

The best results are achieved with the TF-IDF vectorization technique with the parameters:

- max_features: 30000
- min_df: 10
- stop_words: english



model_name	embedding_name	dataset_name	accuracy	recall_macro	precision_macro	f1_macro
svm	tfidf	yelp	0.553211	0.553201	0.550660	0.551523
lr	tfidf	yelp	0.547111	0.547112	0.541973	0.543788
svm	word2vec	yelp	0.536461	0.536363	0.538431	0.536515
svm	fasttext	yelp	0.534737	0.534651	0.535932	0.534653
lr	word2vec	yelp	0.535711	0.535694	0.532544	0.533646
lr	fasttext	yelp	0.532761	0.532726	0.529282	0.530427
svm	glove	yelp	0.519512	0.519505	0.519645	0.518962
lr	glove	yelp	0.517662	0.517655	0.514020	0.515194
nb	tfidf	yelp	0.513287	0.512683	0.519636	0.512396
transformer_embeddings	bert-base-uncased	yelp	0.512162	0.511719	0.511288	0.509700
pretrained_word_embeddings	glove	yelp	0.508637	0.508906	0.503637	0.502083
custom_embeddings	custom_word_embedding	yelp	0.503287	0.503471	0.499518	0.500008
pretrained_word_embeddings	fasttext	yelp	0.493063	0.492983	0.485099	0.486454
rf	word2vec	yelp	0.452614	0.452132	0.443984	0.443424
rf	tfidf	yelp	0.461014	0.460504	0.454652	0.436197
rf	glove	yelp	0.430689	0.430239	0.423961	0.423597
rf	fasttext	yelp	0.430840	0.430282	0.423080	0.421993

Figure 9.5: YELP models results

and classification model SVM with the following parameters:

- C: 1
- kernel: rbf

Looking at the F1 score, the best model performance is decent, at best. From the confusion matrix, we can conclude that not all classes are captured equally. Accuracies are 68%, 48%, 46%, 48%, and 65% for extremely negative, negative, neutral, positive, and extremely positive classes, respectively. From the confusion matrix, is easy to detect that classes "near each other" get often mistaken, while there is far less probability of error if classes are more distant. This suggests that maybe there is a fine line between classes, hence the lower score. The confusion matrix with raw number is shown in Figure 9.6, and the normalized version in Figure 9.7.

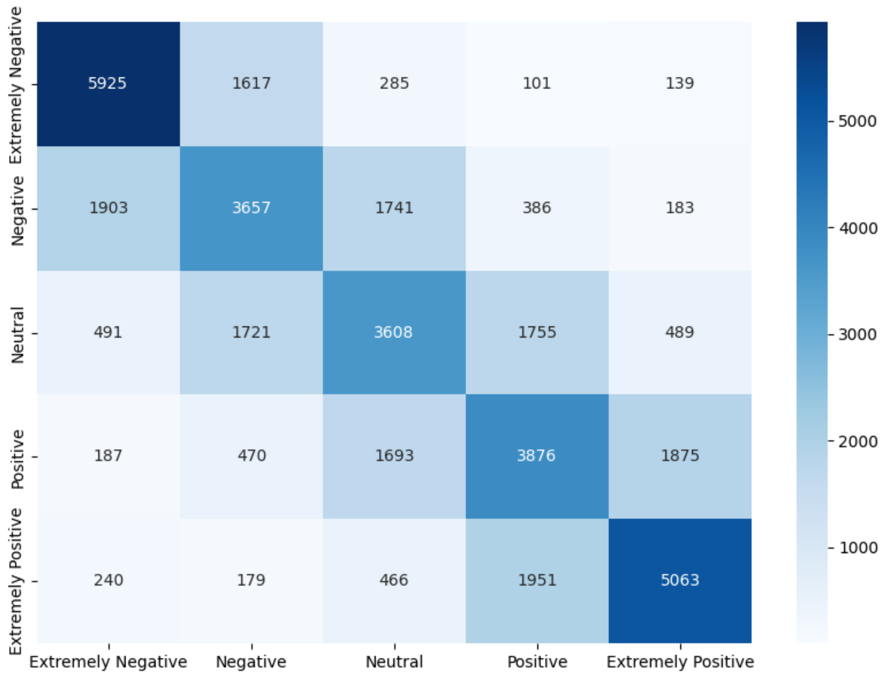


Figure 9.6: Yelp confusion matrix

9.4.3 Corona tweets results

The results for the Corona tweets dataset are displayed in Figure 9.8.
The maximum achieved accuracy is 65% F1-macro score. The second best model is the LR classification model with TF-IDF, with 62% F1-score. After that, there is a huge drop in terms of accuracy, and the worst model is RF with TF-IDF, with 14% F1-score, indicating huge differences in F1-metrics, larger than in the two previous cases.

The best results are achieved with the TF-IDF vectorization technique with the parameters:

- max_features: 30000
- min_df: 10

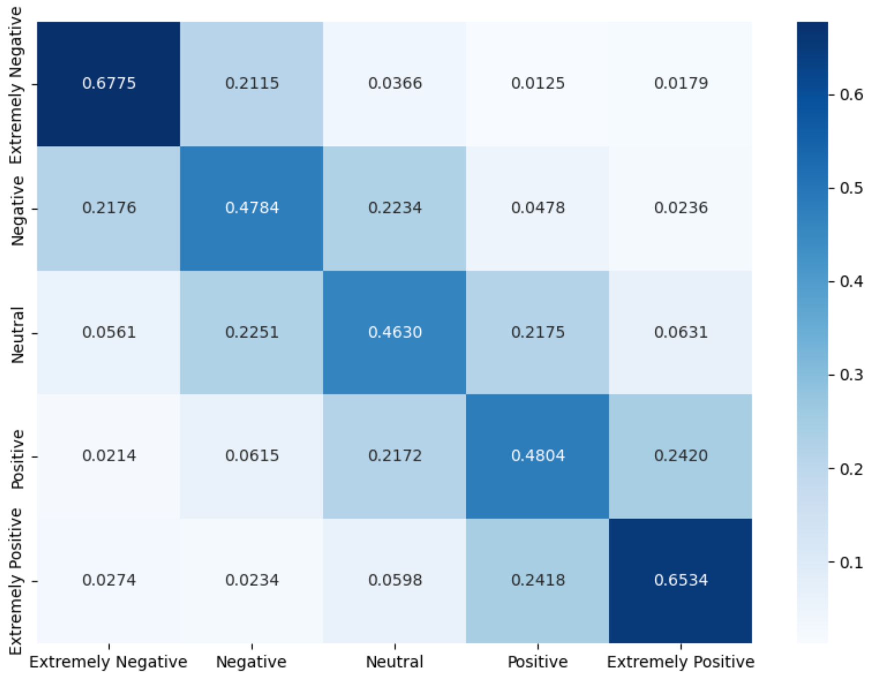


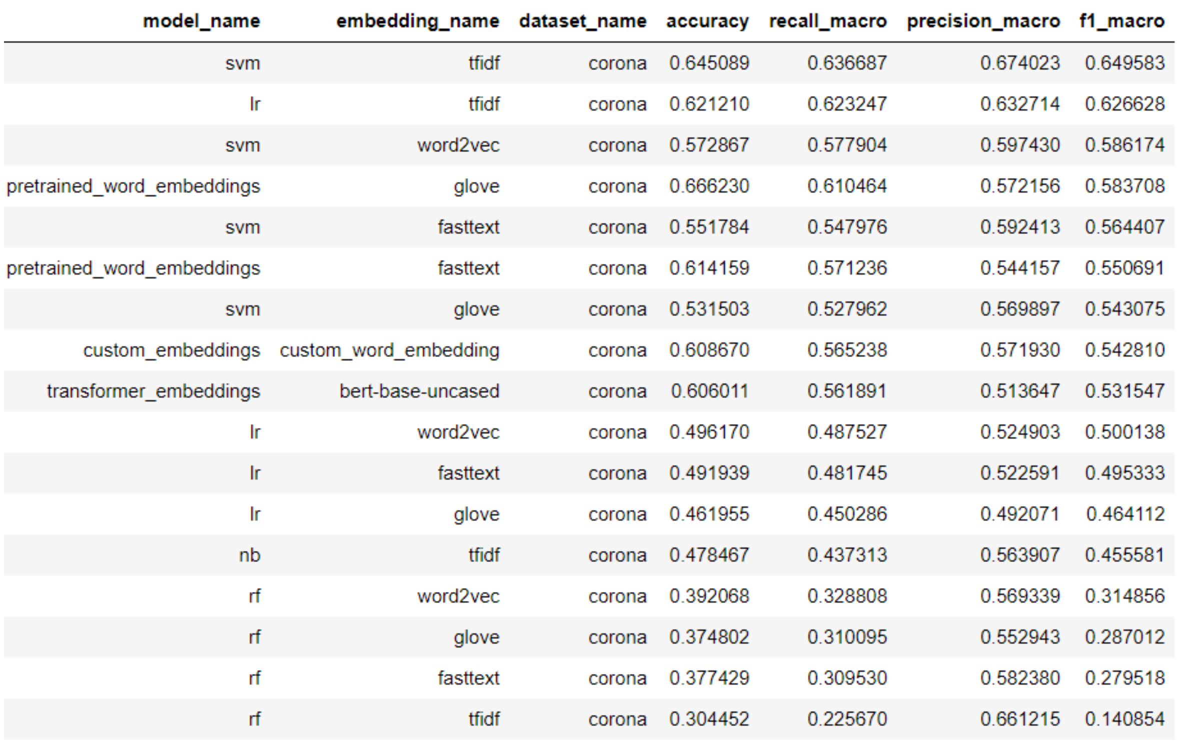
Figure 9.7: Yelp normalized confusion matrix

- stop_words: english

and classification model SVM with the following parameters:

- C: 10
- kernel: rbf

Looking at the F1 score, the best model performance is a bit better than the best model performance for yelp reviews. Accuracies are 73%, 57%, 70%, 60%, and 77% for extremely negative, negative, neutral, positive, and extremely positive classes, respectively. But, as opposed to yelp class scores, classes "near each other" are less likely to be mistaken, but some classes which are more distant, are being mistaken. Negative and positive classes are mistaken in 13% in both cases. This dataset can, in some cases, can



model_name	embedding_name	dataset_name	accuracy	recall_macro	precision_macro	f1_macro
svm	tfidf	corona	0.645089	0.636687	0.674023	0.649583
lr	tfidf	corona	0.621210	0.623247	0.632714	0.626628
svm	word2vec	corona	0.572867	0.577904	0.597430	0.586174
pretrained_word_embeddings	glove	corona	0.666230	0.610464	0.572156	0.583708
svm	fasttext	corona	0.551784	0.547976	0.592413	0.564407
pretrained_word_embeddings	fasttext	corona	0.614159	0.571236	0.544157	0.550691
svm	glove	corona	0.531503	0.527962	0.569897	0.543075
custom_embeddings	custom_word_embedding	corona	0.608670	0.565238	0.571930	0.542810
transformer_embeddings	bert-base-uncased	corona	0.606011	0.561891	0.513647	0.531547
lr	word2vec	corona	0.496170	0.487527	0.524903	0.500138
lr	fasttext	corona	0.491939	0.481745	0.522591	0.495333
lr	glove	corona	0.461955	0.450286	0.492071	0.464112
nb	tfidf	corona	0.478467	0.437313	0.563907	0.455581
rf	word2vec	corona	0.392068	0.328808	0.569339	0.314856
rf	glove	corona	0.374802	0.310095	0.552943	0.287012
rf	fasttext	corona	0.377429	0.309530	0.582380	0.279518
rf	tfidf	corona	0.304452	0.225670	0.661215	0.140854

Figure 9.8: Corona tweets models results

be very problematic, because upon removal of hashtags, only a few words are left, and sometimes that is not enough to make relevant predictions. The confusion matrix with raw number is shown in Figure 9.9, and the normalized version in Figure 9.10.

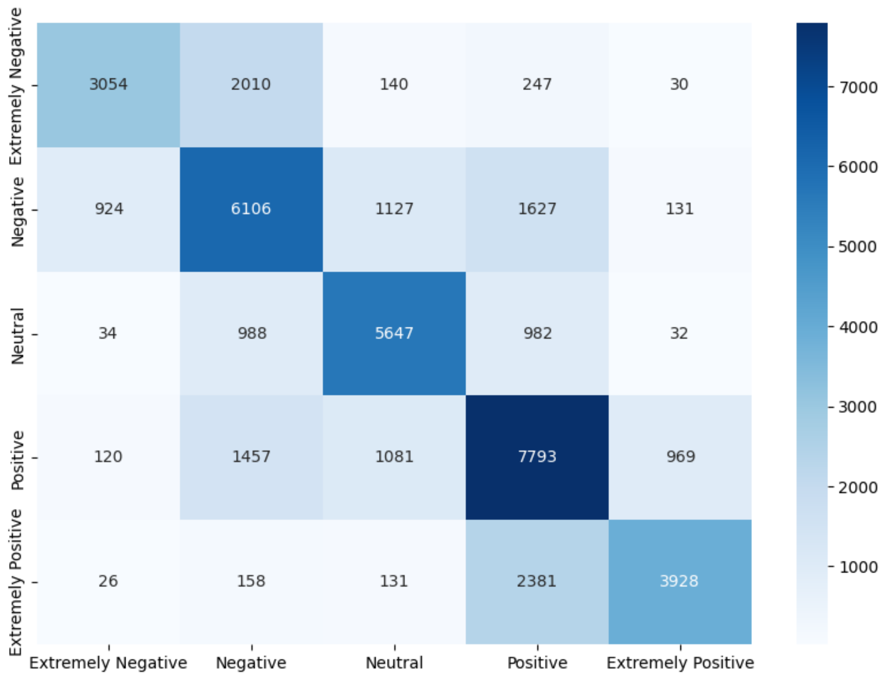


Figure 9.9: Corona tweets confusion matrix

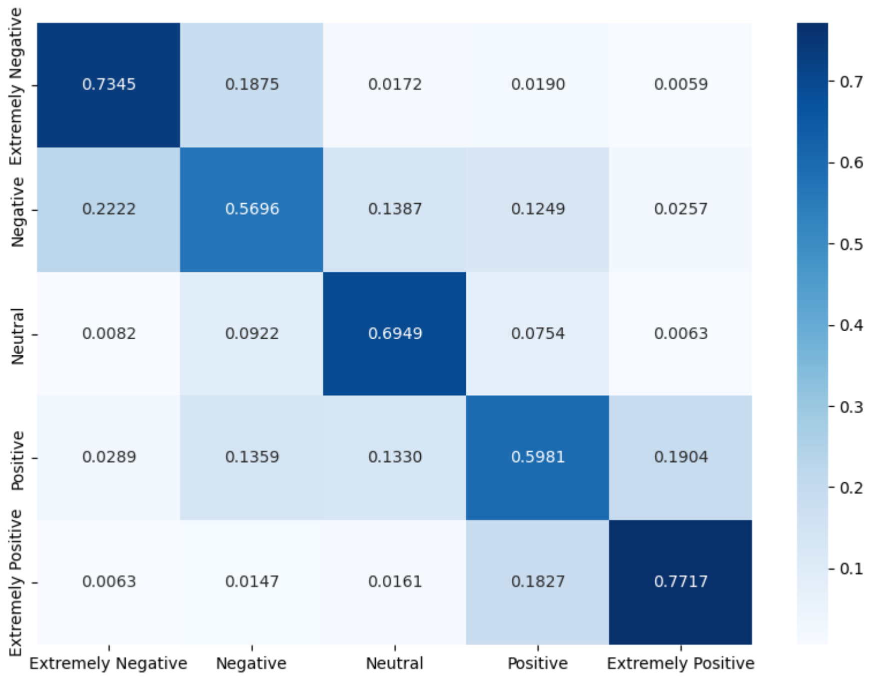


Figure 9.10: Corona tweets normalized confusion matrix

Chapter 10

Conclusions

In this thesis, we tried to unify all features of engineering techniques on the textual data and compare them with machine learning and deep learning algorithms. We started with a bag-of-word approach such as TF-IDF, then move to word embeddings such as Word2Vec, GloVe, and FastText. We finished this with BERT contextual embeddings. From machine learning models we tried SVM, NB, LR, and RF and from deep learning models, we tried LSTM. We provided benchmarks on three different datasets. The goal was to provide an in-depth analysis of different approaches.

These results may come as a bit surprising, based on the previous work, because one of the most naive embedding technique, TF-IDF, in combination with SVM, consistently outperform other embedding techniques and models. Based on the results, one may conclude that deep learning techniques are not necessarily better in comparison to the classical statistical model, and they are trained much faster than deep learning models. Only one preprocessing technique has been used, which is a pipeline with multiple normalization techniques. A different combination of normalization techniques could have been used, something like stemming and lemmatization.

One of the reasons that machine learning models are performing better, is because their hyper-parameters were optimized, were in the deep learning case, they are fixed and all models are trained with them. The reason for this is the computational complexity of training deep learning models. In further work, one can focus only on deep learning hyper-parameter optimizations. Even though machine learning model parameters were optimized with greed

search, the space of parameters was also limited because of training time.
In the case of deep learning hyper-parameter optimizations, there is huge parameter space for someone to configure, starting from the learning rate, LSTM depth, batch size, fine-tuning embedding, and so on.
From transformer models, only BERT embeddings were used in training. Now, there are plenty of different models which can perform better or have the same performance with smaller embedding space, which gives more room for building a more complex network.

Bibliography

[1] Bo Pang and Lillian Lee. *Opinion Mining and Sentiment Analysis.* Foundations and trends in information retrieval, Vol. 2, Nos. 1–2 (2008) 1–135. 2008.

[2] Meena Rambocas and João Gama. "Marketing Research: The Role of Sentiment Analysis". In: *FEP WORKING PAPER SERIES* (2013). URL: `https://wps.fep.up.pt/wps/wp489.pdf`.

[3] Daekook Kang and Yongtae Park. "Review-based measurement of customer satisfaction in mobile service: Sentiment analysis and VIKOR approach". In: *Expert Systems with Applications* (2014). URL: `https://www.sciencedirect.com/science/article/abs/pii/S0957417413006027`.

[4] Federico Neri, Carlo Aliprandi, and Federico Capeci. "Sentiment Analysis on Social Media". In: (2012). URL: `https://ieeexplore.ieee.org/abstract/document/6425642`.

[5] Minqing Hu and Bing Liu. "Mining and Summarizing Customer Reviews". In: *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining* (2004). URL: `https://www.cs.uic.edu/~liub/publications/kdd04-revSummary.pdf`.

[6] Andrea Esuli and Fabrizio Sebastiani. "SentiWordNet: A Publicly Available Lexical Resource for Opinion Mining". In: (2006). URL: `http://www.lrec-conf.org/proceedings/lrec2006/pdf/384_pdf.pdf`.

[7] Yoon Kim et al. "Temporal Analysis of Language through Neural Language Models". In: (2014). URL: `https://arxiv.org/pdf/1405.3515.pdf`.

[8] CHRIS NICHOLLS and FEI SONG. "IMPROVING SENTIMENT ANALYSIS WITH PART-OF-SPEECH WEIGHTING". In: (2009). URL: `https://ieeexplore.ieee.org/abstract/document/5212278`.

[9] Li Dong et al. "IMPROVING SENTIMENT ANALYSIS WITH PART-OF-SPEECH WEIGHTING". In: (2009). URL: https://ieeexplore.ieee.org/abstract/document/5212278.

[10] Lei Zhang et al. "Deep learning for sentiment analysis: A survey". In: *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* (2018). URL: https://wires.onlinelibrary.wiley.com/doi/10.1002/widm.1253.

[11] Yoon Kim. "Convolutional Neural Networks for Sentence Classification". In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing* (2014). URL: https://arxiv.org/pdf/1408.5882.pdf.

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

[13] Zachary C. Lipton. "The Mythos of Model Interpretability". In: *Communications of the ACM* (2016). URL: https://arxiv.org/pdf/1606.03490.pdf.

[14] Ashish Vaswani et al. "Attention Is All You Need". In: (2017). URL: https://arxiv.org/abs/1706.03762.

[15] Jacob Devlin et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: (2018). URL: https://arxiv.org/pdf/1810.04805.pdf.

[16] Maite Taboada et al. "Lexicon-Based Methods for Sentiment Analysis". In: *Computational Linguistics* (2011). URL: https://direct.mit.edu/coli/article/37/2/267/2105/Lexicon-Based-Methods-for-Sentiment-Analysis.

[17] Prem Melville, Wojciech Gryc, and Richard D. Lawrence. "Sentiment Analysis of Blogs by Combining Lexical Knowledge with Text Classification". In: *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 09* (2009). URL: https://dl.acm.org/doi/abs/10.1145/1557019.1557156.

[18] Guang Qiu et al. "DASA: Dissatisfaction-oriented Advertising based on Sentiment Analysis q". In: *Expert Systems with Applications* (2010). URL: https://www.sciencedirect.com/science/article/abs/pii/S095741741000148X.

[19] Hanhoon Kang, Seong Joon Yoo, and Dongil Han. “Senti-lexicon and improved Naïve Bayes algorithms for sentiment analysis of restaurant reviews”. In: *Expert Systems with Applications* (2012). URL: https://www.sciencedirect.com/science/article/abs/pii/S0957417411016538.

[20] María-Teresa Martín-Valdivia et al. “Sentiment polarity detection in Spanish reviews combining supervised and unsupervised approaches”. In: (2013). URL: https://www.sciencedirect.com/science/article/abs/pii/S0957417412013267.

[21] Tomas Mikolov et al. “Efficient Estimation of Word Representations in Vector Space”. In: *Proceedings of Workshop at ICLR* (2013). URL: https://arxiv.org/abs/1301.3781.

[22] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. “GloVe: Global Vectors for Word Representation”. In: (2014). URL: https://nlp.stanford.edu/pubs/glove.pdf.

[23] Piotr Bojanowski et al. “Enriching Word Vectors with Subword Information”. In: (2016). URL: https://arxiv.org/abs/1607.04606.

[24] Liang Chih Yu et al. “Refining Word Embeddings for Sentiment Analysis”. In: *Journal of ICT Standardization* (2017). URL: https://aclanthology.org/D17-1056/.

[25] Sayyida Tabinda Kokab, Sohail Asghar, and Shehneela Naz. “Transformer-based deep learning models for the sentiment analysis of social media data”. In: *Array* (2022). URL: https://www.sciencedirect.com/science/article/pii/S2590005622000224.

[26] Sowmya Vajjala et al. *Practical Natural Language Processing*. O’Reilly, 2020.

[27] Akiko Aizawa. “An information-theoretic perspective of tf–idf measures”. In: *Information Processing and Management* (2003). URL: https://www.sciencedirect.com/science/article/abs/pii/S0306457302000213.

[28] Yoshua Bengio et al. “Neural Probabilistic Language Model”. In: *Journal of Machine Learning Research* (2003). URL: https://www.jmlr.org/papers/volume3/bengio03a/bengio03a.pdf.

[29] Rohit Mundra et al. *CS 224D: Deep Learning for NLP*. [Online; accessed December 17, 2022]. Winter 2019. URL: https://web.stanford.edu/class/cs224n/readings/cs224n-2019-notes01-wordvecs1.pdf.

[30] Rohit Mundra et al. *CS 224D: Deep Learning for NLP*. [Online; accessed December 17, 2022]. Winter 2019. URL: https://web.stanford.edu/class/cs224n/readings/cs224n-2019-notes02-wordvecs2.pdf.

[31] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer, 2008.

[32] David M W Powers. "Evaluation: From Precision, Recall and F-Factor to ROC, Informedness, Markedness and Correlation". In: *Mach. Learn. Technol.* (2010). URL: https://arxiv.org/abs/2010.16061.

[33] *How to construct a confusion matrix in LaTeX?* [Online; accessed 03 December, 2023]. URL: https://tex.stackexchange.com/questions/20267/how-to-construct-a-confusion-matrix-in-latex.

[34] Thomas G. Dietterich. "Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms". In: *Neural Computation* (1998). URL: https://direct.mit.edu/neco/article-abstract/10/7/1895/6224/Approximate-Statistical-Tests-for-Comparing.

[35] Daniel Jurafsky and James H. Martin. *Speech and Language Processing*. Pearson/Prentice Hall, 2008. URL: https://web.stanford.edu/~jurafsky/slp3/4.pdf.

[36] Victor E.Lee, Lin Liu, and Ruoming Jin. *Data Classification: Algorithms and Applications*. Chapman and Hall/CRC. URL: http://www.odbms.org/wp-content/uploads/2014/07/DecisionTrees.pdf.

[37] Bhiksha Raj. *Decision Tree*. [Online; accessed October 20 , 2022]. URL: https://www.cs.cmu.edu/~bhiksha/courses/10-601/decisiontrees/.

[38] Neves Ana et al. *A New Approach to Damage Detection in Bridges Using Machine Learning*. [Online; accessed 10 Feb, 2020]. URL: https://www.researchgate.net/profile/Ana_Neves9/publication/320384373/figure/fig2/AS:682337809469452@1539693419868/The-biological-neuron.png.

[39] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[40] DeepAI. *Perceptron*. [Online; accessed 28 October, 2022]. URL: https://deepai.org/machine-learning-glossary-and-terms/perceptron.

[41] *Long Short-Term Memory (LSTM).* [Online; accessed November 3, 2022]. URL: https://d2l.ai/chapter_recurrent-modern/lstm.html.

[42] Sebastian Ruder. "An overview of gradient descent optimization algorithms". In: (2016). URL: https://arxiv.org/pdf/1609.04747.pdf.

[43] Wikipedia. *Gradient descent.* [Online; accessed 17 December, 2022]. URL: https://en.wikipedia.org/wiki/Gradient_descent.

[44] *Stochastic Gradient Descent on your microcontroller.* [Online; accessed 28 December, 2022]. URL: https://eloquentarduino.github.io/2020/04/stochastic-gradient-descent-on-your-microcontroller/.

Biography

Stefan Dimitrijević was born on the 19th of November 1994 in Leskovac. In 2013 he started his bachelor's studies in Pure Mathematics at the Faculty of Sciences, University of Niš, and finished in 2017. In the same year, he started his master's studies in Data Science at the Faculty of Sciences, University of Novi Sad, where he passed all exams in 2019. He attended ECMI Mathematical Modelling Week in the summer of 2018 where he was included in the project "Modelling effect of time delay for a large network of the seismic monitor".

UNIVERSITY OF NOVI SAD
FACULTY OF SCIENCE
KEY WORD DOCUMENTATION

Accession number:
ANO
Identification number:
INO
Document type: monograph type
DT
Type of record: printed text
TR
Contents code: master thesis
CC
Author: Stefan Dimitrijević
AU
Mentor: Miloš Savić, PhD
MN
Title:
XI
Language of text: English
LT
Language of abstract: English
LA
Country of publication: Republic of Serbia
CP
Locality of publication: Vojvodina
LP
Publication year: 2023.
PY
Publisher: author's reprint
PU
Publishing place: Novi Sad, Trg Dositeja Obradovića 4
PP
Physical description:
(10/99/44/5/45/0/0)
(chapters/pages/references/tables/figures/graphs/additional lists)
PD
Scientific field: mathematics

SF
Scientific discipline: applied mathematics
SD
Key words: Text Analysis, Sentiment Analysis, Natural Language Processing, Machine Learning, Deep Learning
UC
Holding data: Department of Mathematics and Informatics Library, Faculty of Sciences, Novi Sad
HD
Note:
N
Abstract: Three datasets are used for text sentiment detection. Before text is used for detection, it needs to be vectorized. From vectorization techniques, TF-IDF, Word2Vec, FastText, custom embeddings, and BERT embeddings are used. For classification algorithms from machine learning algorithms SVM, NB, RF, and LR are used, and from deep learning algorithms LSTM is used.
AB
Accepted by the Scientific Board on:
10.1.2023.
ASB
Defended:
DE
Thesis defend board:
DB