DEGREE PROJECT IN COMPUTER SCIENCE AND ENGINEERING,
SECOND CYCLE, 30 CREDITS
STOCKHOLM, SWEDEN 2020

Data Augmentation in Solving Data Imbalance Problems

JIE GAO

KTH ROYAL INSTITUTE OF TECHNOLOGY
SCHOOL OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

Abstract

This project mainly focuses on the various methods of solving data imbalance problems in the Natural Language Processing (NLP) field. Unbalanced text data is a common problem in many tasks especially the classification task, which leads to the model not being able to predict the minority class well. Sometimes, even we change to some more excellent and complicated model could not improve the performance, while some simple data strategies that focus on solving data imbalanced problems such as over-sampling or down-sampling produce positive effects on the result. The common data strategies include some re-sampling methods that duplicate new data from the original data or remove some original data to have the balance. Except for that, some other methods such as word replacement, word swap, and word deletion are used in previous work as well. At the same time, some deep learning models like BERT, GPT and fastText model, which have a strong ability for a general understanding of natural language, so we choose some of them to solve the data imbalance problem. However, there is no systematic comparison in practicing these methods. For example, over-sampling and down-sampling are fast and easy to use in previous small scales of datasets. With the increase of the dataset, the newly generated data by some deep network models is more compatible with the original data. Therefore, our work focus on how is the performance of various data augmentation techniques when they are used to solve data imbalance problems, given the dataset and task?

After the experiment, Both qualitative and quantitative experimental results demonstrate that different methods have their advantages for various datasets. In general, data augmentation could improve the performance of classification models. For specific, BERT especially our fine-tuned BERT has an excellent ability in most using scenarios(different scales and types of the dataset). Still, other techniques such as Back-translation has a better performance in long text data, even it costs more time and has a complicated model. In conclusion, suitable choices for data augmentation methods could help to solve data imbalance problems.

Keywords

Data augmentation, Data imbalance, NLP, Deep learning, Comparison.

Sammanfattning

Detta projekt fokuserar huvudsakligen på de olika metoderna för att lösa dataobalansproblem i fältet Natural Language Processing (NLP). Obalanserad textdata är ett vanligt problem i många uppgifter, särskilt klassificeringsuppgiften, vilket leder till att modellen inte kan förutsäga minoriteten Ibland kan vi till och med byta till en mer utmärkt och komplicerad modell inte förbättra prestandan, medan några enkla datastrategier som fokuserar på att lösa data obalanserade problem som överprov eller nedprovning ger positiva effekter på resultatet. vanliga datastrategier inkluderar några omprovningsmetoder som duplicerar nya data från originaldata eller tar bort originaldata för att få balans. Förutom det används vissa andra metoder som ordbyte, ordbyte och radering av ord i tidigare arbete Samtidigt har vissa djupinlärningsmodeller som BERT, GPT och fastText-modellen, som har en stark förmåga till en allmän förståelse av naturliga språk, så vi väljer några av dem för att lösa problemet med obalans i data. Det finns dock ingen systematisk jämförelse när man praktiserar dessa metoder. Exempelvis är överprovtagning och nedprovtagning snabba och enkla att använda i tidigare små skalor av datamängder. Med ökningen av datauppsättningen är de nya genererade data från vissa djupa nätverksmodeller mer kompatibla med originaldata. Därför fokuserar vårt arbete på hur prestandan för olika dataförstärkningstekniker används när de används för att lösa dataobalansproblem, givet datamängden och uppgiften?

Efter experimentet visar både kvalitativa och kvantitativa experimentella resultat att olika metoder har sina fördelar för olika datamängder. I allmänhet kan dataförstärkning förbättra prestandan hos klassificeringsmodeller. För specifika, BERT speciellt vår finjusterade BERT har en utmärkt förmåga i de flesta med hjälp av scenarier (olika skalor och typer av datamängden). Ändå har andra tekniker som Back-translation bättre prestanda i lång textdata, till och med det kostar mer tid och har en komplicerad modell. Sammanfattningsvis lämpliga val för metoder för dataökning kan hjälpa till att lösa problem med obalans i data.

Nyckelord

Dataförstoring, Data obalans, Textklassificering, Naturlig språkbehandling, Djup lärning.

Contents

1 Introduction **1**
- 1.1 Concept . . . 1
- 1.2 Problem . . . 2
- 1.3 Related Work . . . 2
- 1.4 Solution . . . 2
- 1.5 Research question . . . 3
- 1.6 Outline . . . 3

2 Background **5**
- 2.1 Data Imbalance . . . 5
  - 2.1.1 Sampling . . . 6
  - 2.1.2 Evaluation . . . 6
- 2.2 Data Augmentation Background . . . 7
- 2.3 Data Classification Background . . . 10

3 Methods **12**
- 3.1 Task words strategy . . . 12
- 3.2 The Character of Target Words . . . 13
  - 3.2.1 Modifiers . . . 13
  - 3.2.2 Verbs . . . 13
- 3.3 The Number of Target Words . . . 14
- 3.4 Method implementation . . . 15
  - 3.4.1 BERT/Fine-tuned BERT . . . 15
  - 3.4.2 fastText . . . 17
  - 3.4.3 Back Translation . . . 17
  - 3.4.4 Deletion . . . 18
- 3.5 Method Evaluation . . . 18

4 Experiment and Results **20**
4.1 Data . . . 20
4.1.1 The Scale of Training Set . . . 20
4.1.2 Dataset Analysis . . . 21
4.2 Implementation Details of Data Augmentation Models . . . 21
4.2.1 BERT . . . 22
4.2.2 Fine-tuned BERT . . . 23
4.2.3 fastText . . . 23
4.2.4 Back Translation . . . 23
4.3 Implementation Details of Classifiers Models . . . 24
4.3.1 SVM . . . 24
4.3.2 fastText classification . . . 24
4.3.3 Bi-LSTM . . . 24
4.4 Quantitative Analysis . . . 24

5 Discussion and Conclusion **31**
5.1 Discussion . . . 31
5.1.1 Discussion on Result . . . 31
5.1.2 Discussion on Methods . . . 32
5.2 Conclusion . . . 33
5.2.1 Findings . . . 33
5.2.2 Contributions . . . 34

References **37**

A Original results **41**

List of Figures

2.1 The structure of BERT 9
2.2 The process of generating the synonyms of a sentence 10

3.1 The process of training for Pre-trained BERT 16
3.2 How to implement label in Pre-trained BERT structure 16
3.3 The structure of fastText 17

4.1 The distribution of length of sentences of Twitter dataset . . . 21
4.2 The distribution of length of sentences of Ag News dataset . . 22
4.3 The Recall of various methods in Twitter dataset 28
4.4 The F1-score of various methods in Twitter dataset 29
4.5 The Recall of various methods in Ag News dataset 29
4.6 The F1-score of various methods in Ag News dataset 30

List of Tables

2.1 Confusion Matrix 7

3.1 Modify example 13
3.2 Verbs synonyms example 14

4.1 The original dataset 21
4.2 After Data Augmentation 22
4.3 The Recall for every single class of Twitter dataset 26
4.4 The F1-score for every single class of Twitter dataset 27
4.5 The Recall for every single class of Ag News dataset 27
4.6 The F1-score for every single class of Ag News dataset 28

A.1 The F1-score of Twitter dataset in SVM Classification 41
A.2 The F1-score of Twitter dataset in Bi-LSTM Classification . . 42
A.3 The F1-score of Twitter dataset in fastText Classification . . . 42
A.4 The Recall of Twitter dataset in SVM Classification 43
A.5 The Recall of Twitter dataset in Bi-LSTM Classification 43
A.6 The Recall of Twitter dataset in fastText Classification 44
A.7 The F1-score of Ag News dataset in SVM Classification 44
A.8 The F1-score of Ag News dataset in Bi-LSTM Classification . 45
A.9 The F1-score of Twitter dataset in fastText Classification . . . 45
A.10 The Recall of Ag News dataset in SVM Classification 46
A.11 The Recall of Twitter dataset in Bi-LSTM Classification 46
A.12 The Recall of Twitter dataset in fastText Classification 47

Chapter 1

Introduction

1.1 Concept

Natural Language Processing (NLP) is a field of Artificial Intelligent, which aims to make the machine is able to understand and derive information from human natural languages. NLP refers to many tasks such as Machine Translation, Text Categorization, Dialogue System, etc. Natural language is the essence of human history and culture: people from all over the country use their different languages. There are more than 3000 spoken languages in the world, but only hundreds of them have writing systems [1]. Given thousands or millions of people or declarations in given geography (sometimes there are special accents and grammar within the same language system), this is a complex and unmanageable problem previously. Therefore, some previous automatic translation software usually has bad performance when the speech is too long or not very common.

Data is an important resource when building NLP models that the scale and the quality of the data directly affect the performance of models. Due to the limited source of annotated data and privacy issues, the development of both big tech companies and small startups was held up by the availability of the data. And that is the reason why many scientists and researchers are currently working on building large-scale open-source datasets for study and research. For example, the booming development of Image Processing tasks such as Face Recognition, Image Retrieval partially own to the dataset ImageNet.

In this paper, we introduce some methods of Data Augmentation (DA), including how to implement them in NLP tasks.

1.2 Problem

The scope of the paper is how to solve data imbalance problems in text data classification tasks. Data imbalance would lead that models can not learn enough information for distinguishing the classes. In detail, the models tend to assign samples to the majority class if it is trained with imbalanced data. Compare to other types of data such as image, text data is sequential and context-sensitive. According to the paper [2], text data are symbolic, discrete, compositional, and sparse. Even if only one word is changed, the meaning of the whole sentence may be different, so the generated text should be different while keeping the original information. Also, human language has a unique structure for expressing and understanding, so data augmentation still has many constraints.

1.3 Related Work

Creating new images by geometric transformations is a common practice in computer vision [3] [4], which could improve the recognition of handwriting characters by applying transformations to real images. The intensity of RGB (color coding) channels can also be altered by using the principal component analysis to generate new data [5].

Different from computer vision, there are not too many available methods and researches in text augmentation, we have described some reasons in the Problems section 1.2 . In all the related papers and researches, lexical substitution is a common way, which could be found in papers [6], [3], and [7]. There is also other way in the paper [8], which refers to some easy text manipulation, including random insert, random swap or random delete words in a sentences. We modify some of their ideas to our text augmentation.

Expect for some previous solutions, we also propose some new ideas for comparing, to figure out how is the performance for various types of text data. Therefore, this piece of work is an engineering endeavor than academic research. We make tests with different options(parameters, scales...) for different usage of the dataset, and persuasively summarize them.

1.4 Solution

In the report, we choose several common text augmentation methods and compare them with the method with deep learning approaches.

The performance of data augmentation is evaluated by classification tasks. Besides, for a detailed research about the different using scenarios, we split a complete text data into several parts, from 20% to 100%. By this way, we could have a clear and comprehensive comparison for the effect of data augmentation methods.

1.5 Research question

This thesis studies how to apply different data augmentation methods to solve data imbalanced issues and also how is their performance when comparing traditional over-sampling and down-sampling techniques. In detailed, we are trying to answer the following research question:

> How is the performance of various data augmentation techniques when they are used to solve data imbalance problems? For solving this question efficiently, we need to answer
>
> 1. What is the strength of our data augmentation methods compare to the previous methods?
> 2. How to evaluate the result from all those methods?

1.6 Outline

The rest of the thesis is organized as follow:

- Chapter 2 introduces the definition of data imbalance, the background of data augmentation and some methods that is used to evaluate the result of data augmentation. Besides, we also list some previous work on data imbalanced problems.
- Chapter 3 introduces our methods based on previous work. We explain what is our improvement.
- Chapter 4 presents how we process the experiment, including the details such as specific parameters.
- Chapter 5 demonstrates some of our important findings, including a general conclusion, discussion in our experimental process, and future outlook.

Chapter 2

Background

In this chapter, we introduce some basic concepts of NLP and some text augmentation approaches and text classification models. We would introduce their backgrounds, the methods, the reason that we utilize them, and their limitations. This chapter aims to give a basic impression on the knowledge related to text augmentation.

2.1 Data Imbalance

In many real-world applications, a large amount of data is generated with skewed distribution. It reflects an unequal distribution of classes within a dataset, where one specific class has a higher number than others [9][10]. We call the class with more number of samples as the majority class, and the one with a relatively fewer number of samples is called the minority class[10]. If the class distribution is not well balanced, the classifier might get into trouble when searching for the decision boundaries. The worst situation is the classifier predicts all instances as the majority class and ignore the minority class completely.

There are some methods proposed to solve this problem. In general, there are three types of solutions, which are balancing the data distribution, some of the solutions introduces more of the minority class or remove some samples from the majority class, or modifies the class weights, or optimizes the model parameters. These methods have different usage scenarios and sometimes they need to be work together to get a wider range of applications and better performance. For example, optimizing the model parameters is not a good idea when the size of data and class imbalance ratio is extremely high. Hence, a new technique i.e. the combination of sampling method with the algorithm was used [11][12].

2.1.1 Sampling

Over-sampling is a common ways for solving the data imbalance problem previously. Some of popular oversampling methods including Random oversampling (ROS), Synthetic Minority Oversampling Technique (SMOTE), One-sides Sections (OSS), Cluster-based oversampling (CBOS), Wilson's editing (WE), and Borderline-SMOTE (BSM) [13]. Most of these methods re-balance the number of classes by duplicating samples from the minority class. Besides, down-sampling is another solution that removes some samples from the majority class. However, removing the sample can lead to an under-fitting problem. When we do not have enough data, over-sampling is more suitable considering the scale of data needed to converge. Also, simply duplicating original data could lead to the over-fitting problem, so a better measure is to use some smart over-sampling algorithms such as SMOTE, which could generate a sample of the minority class by interpolating two neighborhood samples.

In this paper, we explore some new idea to solve data imbalance problems.

2.1.2 Evaluation

If we want to compare the effect of various DA methods, a proper evaluation methods is necessary. Usually, accuracy is a good evaluation for classification method, but a high accuracy is not a good way when evaluating the performance of a model on an imbalanced dataset. For example, if we have a mailbox and 99% of emails are normal, and the rest is spam. In this case, a classifier without any training could reach a 99% accuracy score by just predicting every email as normal. Therefore, we need to pick up a better evaluation index that could fairly reflect the real performance of prediction for imbalanced data.

We find a solution from confusion matrix, which is illustrated in Figure 2.1 (for binary classification). In the confusion matrix, True Negative is the number of negative examples correctly classified(TN), False Positive is the number of negative examples incorrectly classified as positive(FP), False Negative is the number of positive examples incorrectly classified as negative (FN) and True Positive is the number of positive examples correctly classified (TP).

Recall is the fraction of the total amount of relevant instances that are successfully retrieved and is defined as $Recall = TP/(TP+FP)$, Precision is the fraction of retrieved documents that are relevant to the query and is defined

		Prediction outcome		
		p	**n**	**total**
actual value	**p′**	True Positive	False Negative	P′
	n′	False Positive	True Negative	N′
	total	P	N	

Table 2.1: Confusion Matrix

as $Precision = TP/(TP+FN)$. From Table 2.1, If we set the minority class as the positive class and set the others are negative, we could know if there were more minority class samples is predicted to be positive. If so, we think the classification model has been alleviated to a certain extent. So we choose Recall to do the evaluation, but improving Recall typically reduces Precision and vice versa. That is if the number of false positives decreases, but false negatives increase. As a result, Precision increases, while Recall decreases. For this problem, we choose F-measure that is the harmonic mean of Precision and Recall, and it is defined as $F1 = 2 * Precision * Recall/(Precision + Recall)$.

Finally, we choose Recall to observe the minority class and choose F1-score to observe the whole performance of the model.

2.2 Data Augmentation Background

Data augmentation is a strategy used to improve the diversity and the quality of the data, especially in the computer vision domain. The image is comprised of independent pixels that it can be easily manipulated by adding noise, cropping, padding, or flipping the image while the original information remains. Encouraged by the wide usage of image augmentation in the computer vision area, researchers are working on introducing text augmentation to solve text data imbalance problems. However, in text augmentation tasks, sentences are sequential and words are always highly related to each other.

Even though a tiny modification in the sentence might lead to an overturn of the meaning. For example, if we use flip in the sentence "I love Tom." to "Tom love I", the meaning is completely different. Therefore, a excellent DA method needs to follow language grammars and remain the original meaning when augmenting the sentence. For this purpose, we choose five methods.

- BERT, which stands for a language representation model: Bidirectional Encoder Representations from Transformers. In the paper, BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding [14], the author proposes a language model based on the famous deep learning algorithm, Transformers [15]. Transformers is a language representation model based on attention architecture, which creatively use attention to solve the long term memory missing in the encoder and decoder with the self-attention structure instead of using traditional Recurrent Neural Network architecture. The algorithm of Transformers could decrease the running time because it supports computation in parallel. BERT stands for Bidirectional Encoder Representations from Transformers, and it uses WordPiece embedding with 30000 token vocabularies, also BERT is trained with quite a lot of corpus. which includes the BookCorpus (800M words) and English Wikipedia (2500M words) [14]. Therefore, BERT has a better understanding of natural language than previous models, particularly in conversational search. Because of these, BERT shows its competence in Machine Translation, Sentence Classification, and Sentence Prediction. Also, it provides a flexible way for various tasks, so we could use this advantage instead of re-training another Transformers encoders, Because Pre-trained BERT is already trained by the Masked LM task, We could do the same task based on our datasets directly. In Particular, we use it to generate words synonyms to replace words from sentences, then it becomes a new sample when doing classification.2.1 The picture 2.1 shows the structure of BERT. Inside the structure, we could find how it connected with different components. The first important layer of BERT is its self-attention structure, it inherits from the encoder of Transformers but adds more layers so it has a better understanding with a sentence sequence[14].

  In order to make the replace word could compatible with the meaning of label in some semantic classification task, fine tuning is a beneficial process. Like the picture 2.2 shows, this idea is from the Data Augmentation by Words with Paradigmatic Relations [7], if we add label inside the

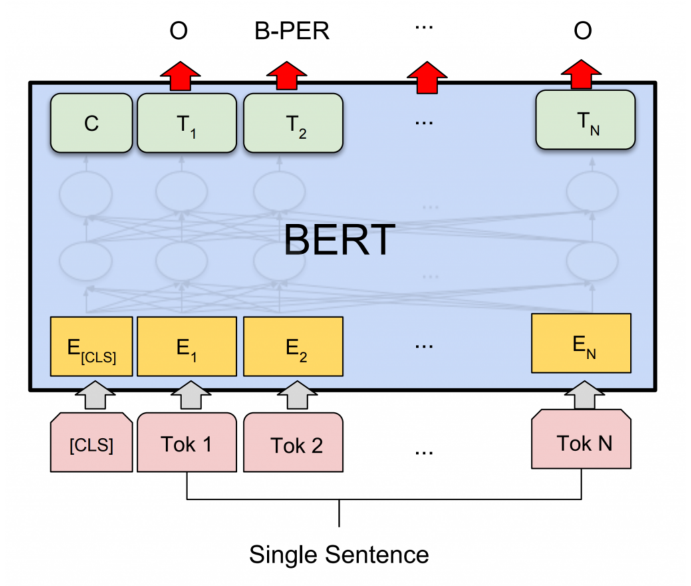

Figure 2.1: The structure of BERT

system and then do augmentation, the new generation will consider the semantics of labels, the generation will be more accurate.

- Generative Pre-Training(GPT-2). Similar to BERT, GPT also uses a part of the Transformers structure, and it uses the decoder of the Transformers structure [16]. Traditional language models need a substantial labeled sample, but sometimes that is unavailable to get so many clean and useful data. In this situation, how to use the model from other domains or how to finish transfer leaning by building a more general model will be the key to solve this kind of problem. GPT-2 Natural language understanding in a wide range of diverse tasks such as question answering, semantic similarity assessment, and document classification. It achieved a better score in some language benchmark scores, like the Global Language Understanding Evaluation benchmark(GLUE)[17].

- fastText is a text classifier algorithm developed by Facebook, which provides a simple and efficient method of text classification and representation learning, with performance comparable to deep learning and faster [18]. Compare to BERT and GPT-2, fastText is not a deep learning structure, it could be used to make word presentation and text classification. In general, fastText includes three parts, the model structure, Hierarchical Softmax, and N-gram features. In data augmentation, we use fastText

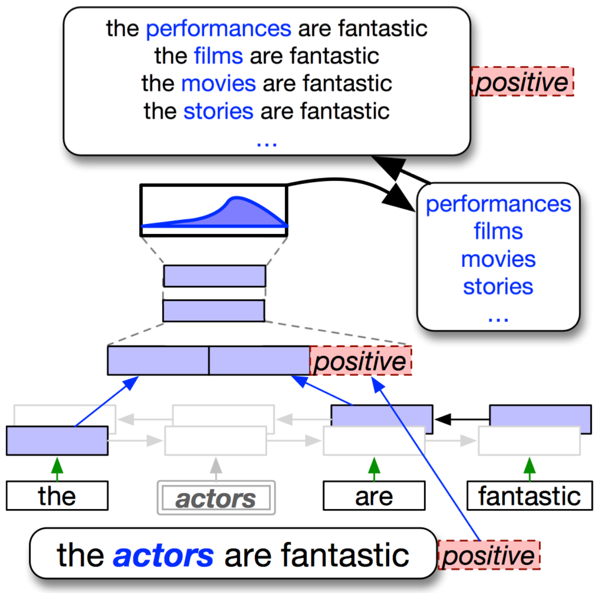

Figure 2.2: The process of generating the synonyms of a sentence

to replace some words inside a sentence. Because fastText could find a synonym of some words. Firstly, use data to fine-tune the word generation model, this step is to help the model to understand the contextual meaning, then the model could make word prediction based on the sentence.

- Easy data augmentation[8].

2.3 Data Classification Background

Text classification is a common task in NLP with broad applications, such as document classification, sentiment analysis, topic labeling, spam detection, and intent detection. For our different datasets, we select three classifiers with different structures.

- LSTM classifier. Long short-term memory[19] [20] (LSTM) is a time-series speech algorithm which was proposed in 1997 and then was extended to solve the information vanish in Recurrent Neural Nets [21]. When training an LSTM model, this structure could help to decrease the vanish of the information as it could store every output in the cell of the LSTM memory block. However, although this structure could keep the long term memory, it still has a limitation when processing long text data. LSTM is often used with other advanced algorithms like the Attention structure to keep more information as more as possible.

- fastText classifier. fastText is to build a binary tree to complete the classification task. Considering the linear and logarithmic models of multiple categories, this greatly reduces the training complexity and the time to test the text classifier. fastText also takes advantage of the fact that classes are not balanced (some classes appear more often than others), and uses the Huffman algorithm to build a tree structure that characterizes classes. Therefore, the depth of the tree structure in which categories appear frequently is smaller than the depth of the tree structure in which categories do not appear frequently, which also makes the further calculation more efficient.

- Support-vector machine (SVM, also support-vector networks) is a popular machine learning algorithm that provides many ways to classification tasks and regression tasks. SVM learns by example to assign labels to objects1. For example, an SVM can learn to recognize fraudulent credit card activity by examining hundreds or thousands of fraudulent and non-fraudulent credit card activity reports[22]. In this thesis, SVM is the only non-deep classification model, and sometimes simple models have better performance.

Chapter 3

Methods

In this chapter, we introduce methods and the whole process we used in the whole research work. The method part includes the strategy that choosing the number of words to replace and delete, how to implement them in the specific method, and other techniques such as open-sources. The process part introduces what is our data augmentation methods, how to use the methods and how to do the evaluation. Both two parts are not separated, they are integrated to each other.

3.1 Task words strategy

For text data, it is made of some words, so we could do some operation in its words, such as word replacement, word swap, word deletion, and word insertion. We mainly focus on word replacement and word deletion. For word replacement, we could synthesize new data based on the original one. More than that, we want the newly generated data could be different from the original one, while still maintaining their original meaning. For word deletion, we change its structure by deleting some words in a sentence, and then we have a new sentence. In a word, the augmented data could increase the diversity of data.

No matter word replacement and word deletion, we need to find some words to delete or replace, we name it target words in this report. There are two steps here, the first step is to choose target words from sentences, and the second step is to find their replacements or delete them. Therefore, before data augmentation, we need to choose the character of target words and the number of target words.

3.2 The Character of Target Words

For our case, we choose modifiers and verbs as the character of our target words.

3.2.1 Modifiers

The character of words could affect the meaning of a sentence. In English or other natural languages, we usually understand a sentence by the following main parts: predicative, predicate, object, attribute and adverbial, etc. In practice, not every dialog or sentence is always made of with all of the parts. Sometimes, we would like to omit some such as an object or add some other words such as modifiers. A modifier is a word/phrase/clause which modifies other words in a sentence, which could provide more descriptions or details to the information of sentences. To be specific, a modifier is either an adjective or an adverb. The adjectives modify the nouns, and the adverbs modify the verbs or the adjectives or the other adverbs. [23]

Table 3.1: Modify example

Original	It is probably not easy to make a worthless film...
modifier 1	It is **probably** not easy to make a worthless film...
modifier 2	It is probably not **easy** to make a worthless film...
modifier 3	It is probably not easy to make a **worthless** film...

From the table 3.1 , we find that a sentence has more than one way to express the same meaning by different modifiers, and this characteristic quite matches with our target words. Therefore, the modifier is a potential part of text augmentation. Besides, the modifier is quite a general character in natural languages, such as adjectives, adverbs(and adverbial phrases and adverbial clauses), and our target words also should be very common in sentences. Based on that, we choose the modifier as a character of target words.

3.2.2 Verbs

Similar to modifiers, the verb (including base form, past tense, past participle, singular present, and 3rd person singular present) is also a very common character, especially for sentiment datasets. We choose the verb not only because it widely exists in natural speeches, but it could be the keyword character that affects the label of text datasets.

Table 3.2: Verbs synonyms example

Original	It is not easy to **make** a worthless film...
Replace verb 1	It is not easy to **create** a worthless film...
Replace verb 2	It is not easy to **find** a worthless film...
Replace verb 3	It is not easy to **see** a worthless film...

In the example 3.2, we usually think the original sentence has negative sentiment because of the word 'worthless', which is usually regarded as a negative emotion in human languages, so we could generate a similar verb with a similar sentiment. By doing this, we could produce many new sentences.

Apart from modifiers, verbs is also a character that has an impact on the sentiment of natural language. For example, some verbs such as "hate" and "love", "like" and "dislike" could convey opposite emotions.

In conclusion, since modifiers and verbs could affect the sentiment of sentences, we could use them to generate more data. Meanwhile, they are also a risk that we could change the sentiment of original instances. For this problem, our measures are to add a process of fine-tuning for the specific sentiment. Before generation, the corresponding model would learn the context of sentences, to understand how is sentiment. About how to learn the context, we would introduce further in the following parts.

3.3 The Number of Target Words

Apart from the character of target words, we also need to consider the number of target words, that is because it could affect the quality of the data augmentation. If we change too few words, newly generated sentences do not have not too many differences from the original one, which is probable that the classification model could not distinguish them, and this is a meaningless data augmentation result. If we change too many words, there is another risk that the original sentences change to totally different sentences while losing important information. Therefore, this number should consider both the length of sentences and the target(proportion) of data augmentation.

For having a good data augmentation result, we test different numbers (from two to five words) in the first dataset and five different numbers (from two to six words) in the second dataset.

3.4 Method implementation

After knowing the character of target words and the number of target words, the next step is to generate synonyms by the following models.

There is a way to find synonyms from a fixed word dictionary like WordNet, but this way is not so intelligent because words are quite complex and they have different meanings in different contexts. For example, " The street is so huge, and I am almost lost myself. Unfortunately, I lost my key as well." There are two "lost" in the sentence and they do not share the same meaning. If we use the same synonym to do the replacement, we probably lose the contextual meaning more or less. Therefore, our solution is to do synonyms generation with deep learning models.

3.4.1 BERT/Fine-tuned BERT

As we have introduced the concept and principle of BERT in the Background part, we focus on explaining how to apply BERT and Fine-tuned BERT techniques to data augmentation in this part. BERT was pre-trained by two kinds of unsupervised tasks, which are Masked LM, and Next Sentence Prediction (NSP). In the training process of Masked LM, it masks 15% of all Word-Piece tokens in each sequence at random, and the model will be trained to predict the masked words with those unmasked tokens [14]. We retrain this process again in the experiment. In BERT, we use a different strategy to choose mask tokens instead of random choices. We only mask words with the specified part of a sentence, which are modifiers or verbs. If the i-th token is the target word, we replace the i-th token with (1) the [MASK] token 80% of the time (2) a random token 10% of the time (3) the unchanged i-th token 10% of the time. Then, the model predicts with cross-entropy loss.

In Fine-tuned BERT, we add an extra process to fine-tune the prediction. Expect for the same mask tokens process, we also add the idea proposed in the paper [7]. Because the BERT is not designed for data augmentation, it predicts words without considering the word probability at a position i, i.e., cloze sentence $S\{w_i\}$ and the probability of prediction is $p(.|S\{w_i\})$. We could select z substitutes for the word w_i from a computed probability distribution. However, the prediction is not always compatible with the sentence's label

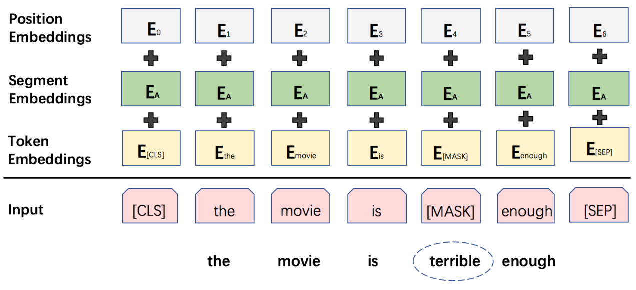

Figure 3.1: The process of training for Pre-trained BERT

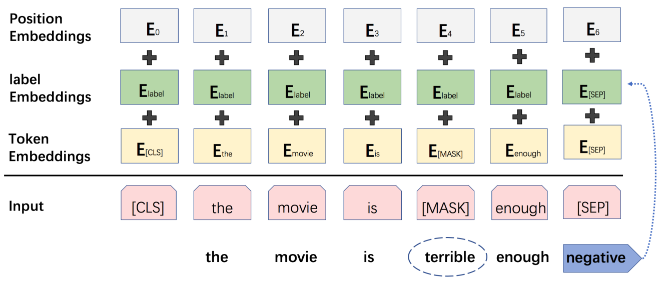

Figure 3.2: How to implement label in Pre-trained BERT structure

that the substitutes with the highest probability might be opposite to the label of the sentence. In this case, data augmentation will damage the sentiment classification tasks. For example, there is a negative review comment "This movie is bad enough". If we use BERT or another language model to predict the substitute for the word $good$, it happens we will get positive words such as $good$ or $nice$ in the top rank probabilities distributions.

The solution proposed in the paper [7] [24] is to alter Language Model to a label-conditional Language Model. We calculate the probability of substituted for the word w_i as $p(.|y, S\{w_i\})$ rather than $p(.|S\{w_i\})$. With the help of the idea, we fine-tune the pre-trained BERT with the label of our data. The pre-trained input representation construction and the fine-tune model architecture are shown in the Figure 3.1 and Figure 3.2 respectively. From these two figures, it's easy to find Fine-tuned BERT could combine the information both the label and the context of a sentence.

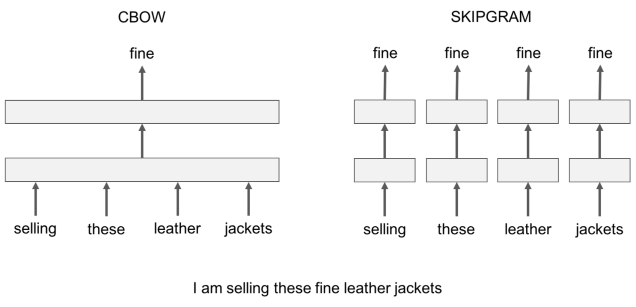

Figure 3.3: The structure of fastText

3.4.2 fastText

fastText is a library for the learning of word embeddings and text classification, which allows one to create unsupervised learning or supervised learning algorithms for obtaining vector representations for words [25]. fastText provides two ways for computing the word representations as shown in the figure 3.3: skip-gram and cbow (continuous-bag-of-words). Skip-gram predicts the center word by surrounding context words, and the cbow predicts surrounding words by the center word. Either skip-gram or cbow, they help the model to represent words dynamically, and in this way, we could find a similar word representation for target words. In our experiment, we choose the cbow model to do the word representation.

Firstly, we use help data that is trained with Wikipedia 2017, UMBC webbase corpus, and statmt.org news dataset (16B tokens). Help data is used to make the fastText model has a general understanding of human language, we use our datasets to compute word vector. In theory, the help data should be related to our dataset, because the help data decide the word vector directly, and the word vector is the only way to find synonyms in fastText. After the training with help data, we could get the nearest neighbors with target words, and choose those neighbors with the highest probabilities.

3.4.3 Back Translation

For an overall comparison, we also use back translation to do data augmentation, though this way is much time costly and low efficiency. Because we do not have enough corpus with the label, it is impossible to train the data augmentation with our data. For solving this problem, we choose Fairseq as a replacement.

Fairseq is a sequence modeling toolkit that allows researchers and developers to train custom models for translation, summarization, language modeling, and other text generation tasks [26]. As it has the pre-trained translation model, and we choose German to English and English to German as the pair language for our back translation task.

3.4.4 Deletion

The method above is more about keeping the same amount of words with the original data because the same words mean a similar structure and few differences. But word deletion also could be an option to do the data augmentation, if we delete some specific words, such as stop words, punctuation, etc. So we also do word deletion to do text augmentation.

Stop words and punctuation usually have limited affection to the sentiment of sentences, so we choose them as target words for word deletion. Firstly we find all the target words of a sentence and delete some of them depend on the length of a sentence, which could be regarded as the diversity of length. To find the target words, we use the library NLTK (Natural Language Toolkit) to select target words.

3.5 Method Evaluation

We evaluate the performance of different text augmentation methods in text classification tasks. We employ three classifiers, which are SVM, LSTM, and fastText classifier.

We execute the classification tasks by the following process.

- Splitting the data into three parts, training dataset, validation dataset, and test dataset.

- Optimizing augmenters and augmenting the text in the training set only using different approaches.

- Collecting the original training dataset and augment training data. Training the classifiers with original and augmented data respectively and validating the result by the validation set. Be aware that we only do data augmentation on the training set instead of the validation set.

- Comparing the classification results between when using original data and augmented data, and comparing the performance of different data augmentation methods.

In the whole classification process, the only difference is the origin of the training dataset, so the final result is convincing and clear.

Chapter 4

Experiment and Results

There are two sections in our whole experiment: text augmentation and text classification. For text augmentation, we have two ways: word deletion and word replacement. Word deletion is to delete target words according to the length of sentences, and word replacement is to replace target words with newly generated words.

After the text augmentation process, we continue to evaluate its result. We choose three classification methods including SVM, Bi-LSTM, and fastText to test the result in various data augmentation situations such as the scale of the dataset. We introduce the specific parameters in the augmentation and classification process in the following part.

4.1 Data

4.1.1 The Scale of Training Set

As our research field is related to solving data imbalance problem, we choose the data from some already existing imbalance datasets, not create them artificially. They are usually generated by human activities, like social networks, business activities, etc. Finally, we choose Twitter dataset and Ag News dataset.

- Twitter [27]. The data is from the international workshop, which is an ongoing series of evaluations of computational semantic analysis systems, organized under the umbrella of SIGLEX.

- Ag News [28]. This is a public dataset which is a collection of more than 1 million news articles. Also, we did not use all the data but part

Dataset	Number of Classes	Classes ratio	Training Set Size	Validation Set Size
Twitter	3	7:6.5:1	28965	7242
Ag News	4	5:1:5:5	75611	14400

Table 4.1: The original dataset

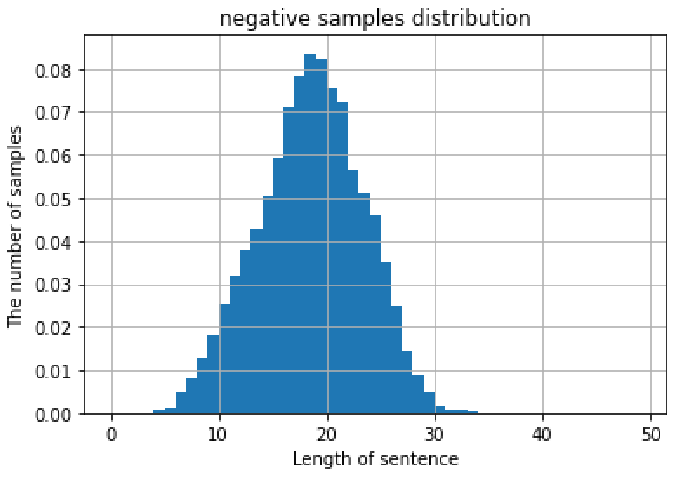

Figure 4.1: The distribution of length of sentences of Twitter dataset

of imbalanced classes.

4.1.2 Dataset Analysis

From Figure 4.1, we could find the class distribution of two datasets. The length of Twitter sentences is concentrated around 20 words, and the distribution is conformed Norm-Distribution. Much difference to Twitter sentences, the length of sentence of Ag News data is much longer figure 4.2, and most of the length is more than 40 words. We choose these two different lengths is to make our research more robust for a diverse dataset.

4.2 Implementation Details of Data Augmentation Models

We choose BERT, fastText, and back translation to do the data generation problems. According to previous parts, we explained the reason why we

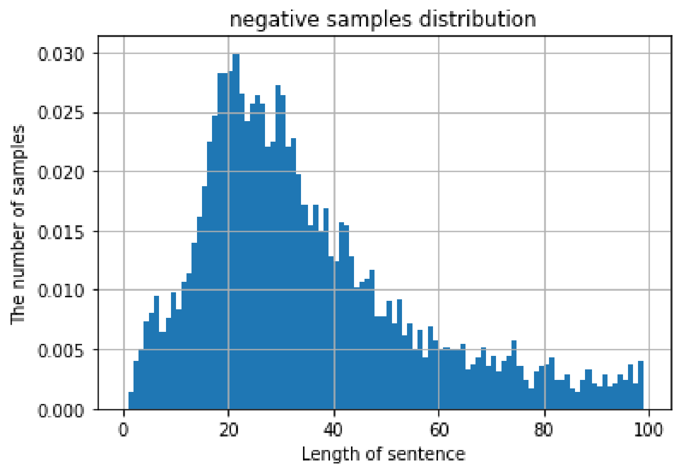

Figure 4.2: The distribution of length of sentences of Ag News dataset

Dataset	Number of Classes	Classes ratio	Training Set Size	Validation Set Size
Twitter	3	7:6.5:7	40817	7242
AG News	4	5:5:5:5	127545	14400

Table 4.2: After Data Augmentation

choose the verb and modifiers as the target words in the word replacement method, and we also explained how we choose the target words to do data augmentation in the Word deletion method. This part refers to how to experiment with every single step.

All the experiments are implemented on PyTorch. To compare the various scale of data affect the result of data augmentation, we split the training data into various rations, which are 20%, 40%, 60%, 80%, 100%, and then do text augmentation respectively. After text augmentation, the minority class has the same amount as the majority class. We present the details of the original datasets table 4.1, and the details of the dataset after data augmentation Tabel 4.2.

4.2.1 BERT

Firstly, We find the target words with NLTK, and then we replace target words with a special character "[MASK]". This is the step for tracking target words, and we use the basic BERT model in this case. 12 hidden layers, 12 attention heads for each attention layer in the encoder part, and the dropout probability are 0.1 and the vocabulary size of the BERT model is 30522. The replacement rate is less than 20 percent of a sample, which is from two to six words in different experiments but we control the whole replacement rate is under 20 percent of a length of sentence, to make sure there is no big change in the meaning.

4.2.2 Fine-tuned BERT

We optimize the original BERT model for our case to make it adapt to our task. To make the comparison equal, we use the same BERT parameters, including 12 hidden layers, 12 attention heads, and the vocabulary size is 30522 as well.

We use the same architecture with pre-trained BERT, but we make some fine-tuning in two parameters for our data. Throughout all the experiments of fine-tuning, we use Adam optimizer with learning rate 5*e-5, set the batch size as 32, and train the model with 20 epochs.

4.2.3 fastText

In the fastText model, we conduct a word representation process, for having a contextual understanding model, with the following parameters. The word embedding dimension is 150, and the subwords are set to 3 words, and the training process is trained by skip-gram algorithm. We set the learning rate is 0.05 after many tests. In the generation process, we set the maximum length of the sentence is 20 words, because fastText words training is based on skip-gram algorithm that has a limited performance to long term memory process, 20 words are many enough to it.

4.2.4 Back Translation

We use the pretained translation library Fairseq to finish the back translation process. In detail, we translate our text data from English to Germany and backward. For each sentence, we generate six candidate translation sentences and then we abandon the most similar one (highest probability).

4.3 Implementation Details of Classifiers Models

4.3.1 SVM

Because our word embedding dimension (feature number) is 100000, so we choose linear kernel to train this SVM model. We set the tolerance for stopping criteria is 1e(-4), set the number of iterations is 1000.

4.3.2 fastText classification

The fastText classification is trained with the following parameters. Its learning rate is 0.2, the epoch is 25, the word n-gram is 2 and the loss calculation method is hierarchical softmax instead of regular softmax.

4.3.3 Bi-LSTM

For the Bi-LSTM classification model, the embedding size is 200, the hidden size is 128. We choose a single layer for this LSTM model, and the mini-batch size is 64, bi-direction LSTM, epoch number is 50, and the learning rate is 2e-5.

4.4 Quantitative Analysis

We compare the performance of over-sampling, down-sampling and our best data augmentation method on both two datasets. These figures 4.3, 4.4 4.5 4.6 show how the quantitative evaluation results change with various scales of dataset.

In general, we could find the performance which includes both Recall and F1-score are improved, almost in all DA methods except for down-sampling in some cases. In those three classification models, SVM classification is one of the best overall performance methods and one of the fastest methods A.1 A.4 for Twitter dataset and Ag News dataset A.7 A.10, so we choose its F1-score in Figure 4.4 and Recall in Figure 4.3 for further analysis. Similarly, we choose the best classifier for Ag-News dataset and compare its F1-score in Figure 4.6 and Recall in Figure 4.5.

For each dataset, DA has a different performance. In the Twitter dataset, we find the Recall of SMOTE and Fine-tuned BERT have similar performance($\pm 1\,\%$), when the scale of the dataset is less than half. That is because our deep network model such as BERT and Fine-tuned BERT still keeps under-fitting and non-contextual understanding with not enough training data. We could valid this idea by increasing the scale, with the increasing of the dataset, Fine-tuned BERT has much more advantage in Recall, which is about higher 3% than other methods when the scale reaches to 100%, while the F1-score is increasing before the scale coming to 50% (showing in Figure 4.3).

While the data augmentation methods do not have the same performance in the Ag News dataset. In this dataset, back translation has best overall score compare to BERT, Fine-tuned BERT and fastText method according to the Figure A.10 and Figure A.7, so we select back translation to compare with traditional methods.

For showing clearly, we make a comparison in Figure 4.5. It is easy to find that back translation has a clear advantage in Recall, which is 2.5% higher than SMOTE when the scale is 20%, and this advantage is downing as the scale of the dataset increases. Similar to the Twitter dataset, the deep model needs to rely on enough datasets to optimize its model, and Fairseq is a pre-trained model so it has congenital advantages compare to other models. Also, because we do not optimize this model with our dataset as there is no proper corpus, which leads to this method has higher performance compared to other methods in the beginning, while this advantage decrease with the increasing of the training dataset. (showing in Figure 4.6).

About F1-score, there is very slight change between SMOTE and Fine-tuned BERT ($\pm 1\,\%$) in both two datasets in Figure 4.4 4.6, but they all have improvement ($4\,\%$-$6\,\%$) compare to original dataset.

Meanwhile, compared to other methods, down-sampling and word deletion have worse results in improving the performance of text classification, either in Recall or F1-score. What's more, its result is even worse than the original dataset on Figure 4.6 and Figure 4.3. For down-sampling, the main reason is that it decreases too many majority classes (about 80%) samples, which probably leads to too much information loss. Therefore, after down-sampling, though we have a balanced dataset, the model is easy to be under-fitting if we train a model with it.

What's more, we also compare the effect on every single class. For example, Table 4.3 shows how changes of Recall in every single class on the Twitter dataset. If the scale is 20% of the dataset, the minority class (named '0') has a Recall score 7.97%, but it increases to 25% after DA, similarly, the performance of using other scales of the dataset are improved as well. Apart from that, it increases the performances of minority classes(over 300%) while keeping the Recall of other class drops less than 10%. Similarly, the F1-Score Table 4.4, and Ag News dataset also presents this trend in Table 4.5

The Recall for Twitter dataset						
	Before DA			After DA		
Label / data	0	1	2	0	1	2
20%	7.97%	66.63%	73.00%	25.16%	65.84 %	66.56%
40%	9.71%	69.03%	72.84%	29.57%	64.12%	71.87%
60%	12.75%	69.39%	73.41%	34.78%	63.94%	72.10%
80%	14.93%	70.33%	72.84%	36.96%	64.99%	72.05%
100%	15.51%	70.67%	73.94%	40.00%	64.66%	73.00%

Table 4.3: The Recall for every single class of Twitter dataset

and Table4.6.

The F1-score for Twitter dataset						
	Before DA			After DA		
Label / data	0	1	2	0	1	2
20%	13.46%	66.81%	67.89%	26.89%	64.85%	67.59%
40%	15.91%	68.34%	68.89%	29.91%	66.36%	69.06%
60%	20.02%	68.89%	69.52%	32.88%	66.66%	69.57%
80%	22.99%	69.33%	69.67%	35.37%	67.26%	69.91%
100%	23.49%	69.93%	70.55%	37.02%	67.43%	70.68%

Table 4.4: The F1-score for every single class of Twitter dataset

The Recall of Ag News dataset								
	Before DA				After DA			
Label / data	0	1	2	3	0	1	2	3
20%	90.42%	97.47%	57.37%	92.26%	89.74%	97.68%	71.84%	90.53%
40%	91.21%	97.79%	61.05%	93.00%	90.79%	97.95%	71.84%	91.68%
60%	91.21%	97.95%	62.89%	93.26%	90.89%	97.95%	69.21%	93.11%
80%	91.84%	98.00%	65.26%	93.53%	91.58%	98.11%	70.00%	92.84%
100%	92.00%	98.21%	65.26%	93.79%	91.53%	98.00%	68.95%	92.95%

Table 4.5: The Recall for every single class of Ag News dataset

The F1-Score of Ag News dataset								
	Before DA				After DA			
Label / data	0	1	2	3	0	1	2	3
20%	91.21%	95.10%	66.06%	91.40%	91.30%	95.23%	71.09%	91.51%
40%	91.91%	95.72%	68.74%	92.22%	92.05%	95.93%	71.75%	92.39%
60%	92.06%	96.20%	69.68%	92.29%	92.25%	96.30%	71.76%	92.67%
80%	92.43%	96.53%	71.68%	92.70%	92.40%	96.70%	72.50%	92.65%
100%	92.61%	96.76%	71.78%	92.89%	92.33%	96.63%	71.68%	92.75%

Table 4.6: The F1-score for every single class of Ag News dataset

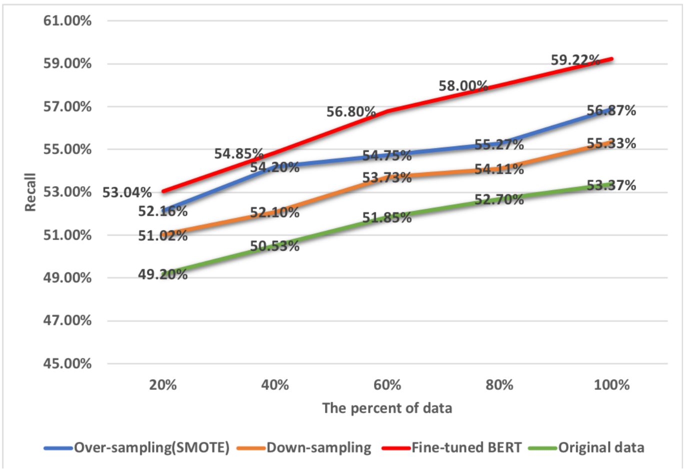

Figure 4.3: The Recall of various methods in Twitter dataset

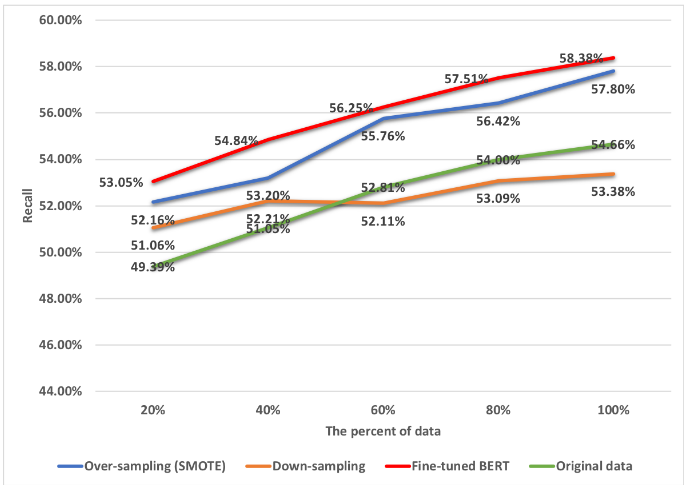

Figure 4.4: The F1-score of various methods in Twitter dataset

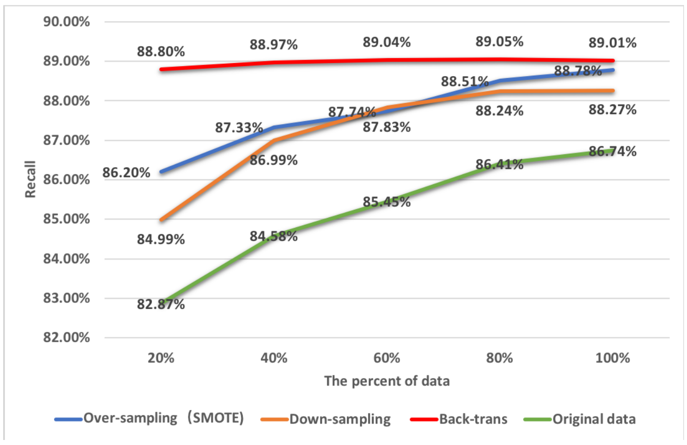

Figure 4.5: The Recall of various methods in Ag News dataset

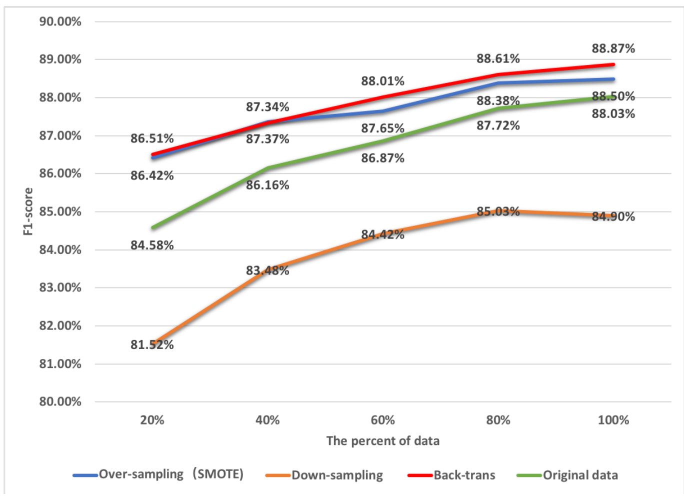

Figure 4.6: The F1-score of various methods in Ag News dataset

Chapter 5

Discussion and Conclusion

In this chapter, we report the discussion, draw conclusions, and demonstrate the contribution of our improvement to this field.

5.1 Discussion

5.1.1 Discussion on Result

Our research is to compare the performance of various data augmentation techniques when they are used to solve data imbalance problems. We want to find the most effective data augmentation methods to get the best result in the NLP field, given dataset and task. According to the final result, we could conclude that the deep network model could increase the performance of models for the classification task. But we also find that some non-machine learning methods such as over-sampling have still its advantage. For example, In the Twitter dataset, we find the Recall of SMOTE and Fine-tuned BERT have similar performance($\pm 1\,\%$), when the scale of the dataset is less than half in given dataset, while BERT needs much more time to train and generate new samples. Therefore, there is no perfect solution for all scenarios, a better solution is to choose the methods by the dataset and tasks.

The reason why we choose different kinds of augmentation methods in the thesis work is that data augmentation in NLP is still a new field, and there is no clear theory about which path is right although some of them have better performance than others in practice. Therefore, we want to compare various methods, including pre-train models such as BERT, widely used word embeddings library fastText, word deletion in sentences, and back-translation library. However, what is out of our expectation is that even a deep model with contextual information of the original sentences cannot always make a difference in the diversity of a sentence, at least the final result is not always better than other methods. If we check the F1-score, we find there is very slight change between SMOTE and Fine-tuned BERT ($\pm 1\,\%$) in both two datasets in Figure 4.4 4.6, but they all have improvement ($4\,\%$-$6\,\%$) compare to original dataset. There are two possible explanations for that, firstly, because the number of task words for replacement is too little to change the meaning very much even we set it to 20% maximize. The second is that we have a mistake in the concept of data augmentation in NLP. As most of our work is based on how to produce other sentences that have a similar meaning to the original sentences, this is a practical experience but lack of theory support.

According to the result, our work is not enough to make a very convincing conclusion, and we still have some work to do in the future.

- Choose more other classification models, minimize the effect from some characters of sentences, like the length and type of sentences.
- Try other characters of a word. Because the character has different effects on different types of sentences, like verbs to sentiment sentences, we could find more other characters to complete the text replacement method.
- For back translation, we would like to train a translation model with our datasets. In that way, our model could have more contextual information about our data.
- If possible, we would like to do more research on the definition of data augmentation and try more NLP tasks. As we mentioned before, all of our work at present is based on data generation/deletion models for the classification task, and other tasks like text summarize dialogue system. etc, there should have more general solutions for them, so we would focus on it in our next step.

5.1.2 Discussion on Methods

Our target is to compare the performance of various data augmentation methods in solving data imbalance problems. For this purpose, we split the target into two parts, augmentation and evaluation, and execute different models between them. The whole process is rigorous and clear.

Nevertheless, data augmentation is not the only way. Using K-fold Cross-Validation, ensembling different re-sampled datasets, and designing a cost function also could solve that problem to a certain extent. In practice, there is no best approach or model suited for all cases and it is strongly recommended to try different techniques and models to evaluate what works best. It is common to combine different approaches.

There are some steps that could be explored more when processing data. Firstly, we use three strategies in processing text sentences, which are word replacement, word deletion, and back-translation. In fact, there are also other ways to augment text data, such as word insertion and word swap, and it is possible that word insertion could have better performance in some cases. Secondly, the target word is also not fixed, and it could be matched with the tasks. For our case, we assume that some characters of sentences have more information than others in English based on some research papers [29] [23]. And then we make some tests to choose the characters. Therefore, even our strategies work well in English, we cannot ensure it has the same performance in other languages.

5.2 Conclusion

In this master thesis project, we mainly pay attention to the comparison between deep learning methods and traditional methods in solving the problem of data imbalance. We choose various ways to complete the process of data augmentation, including substitution, deletion, and sentence translation. After these data augmentation, the imbalanced data is balanced for every single class. We evaluate the result by classification tasks in different scales of datasets and different kinds of datasets.

5.2.1 Findings

We summarized the findings from Part 4 as follows:

- Data augmentation could improve the performance of classification models, especially for the minority class, which could broaden a new approach to the data imbalance problem.

- For complex speech information, back translation shows better performance than word replacement, but it needs much more time costing.

- BERT especially to fine-tune BERT is better than deletion and fastText in general.
- Over-sampling such as SMOTE shows quite robust ability, in both two datasets. Because its model is explainable and easy to understand, it still has many potential using scenarios in many fields. However, down-sampling shows bad results, and one of the reasons is probably because our dataset is huge for it. In this case, down-sampling loses too much information.

5.2.2 Contributions

We summarized our contributions as follows:

- Propose a new method to explain what kind of character of words should be chosen as target words.
- Combine some excellent language model with our data augmentation tasks.
- Make a comprehension comparison about different data augmentation models with various datasets, and then make a clear analysis with their advantages and disadvantages.

In a word, our research focuses on the advantages of the deep network model in solving the problem of data imbalance. After the experiment, we find that Fine-tuned BERT and back-translation methods could improve the whole performance of imbalanced data especially in large scale of dataset, including Recall and F1-score. However, for a small scale of the imbalanced dataset, the previous re-sampling methods could also achieve the same performance with less time cost.

References

[1] S. W. Samaranayake and R. Charoenwisal, “The nature of human language and its characteristics.”

[2] Y. Goldberg, “Neural network methods for natural language processing,” *Synthesis Lectures on Human Language Technologies*, vol. 10, no. 1, pp. 1–309, 2017.

[3] C. Coulombe, “Text data augmentation made simple by leveraging nlp cloud apis,” *arXiv preprint arXiv:1812.04718*, 2018.

[4] T. M. Ha and H. Bunke, “Off-line, handwritten numeral recognition by perturbation method,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 5, pp. 535–539, 1997.

[5] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[6] X. Zhang and Y. LeCun, “Text understanding from scratch,” *arXiv preprint arXiv:1502.01710*, 2015.

[7] S. Kobayashi, “Contextual augmentation: Data augmentation by words with paradigmatic relations,” *arXiv preprint arXiv:1805.06201*, 2018.

[8] J. W. Wei and K. Zou, “Eda: Easy data augmentation techniques for boosting performance on text classification tasks,” *arXiv preprint arXiv:1901.11196*, 2019.

[9] S. Wang and X. Yao, “Multiclass imbalance problems: Analysis and potential solutions,” *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 42, no. 4, pp. 1119–1130, 2012.

[10] N. V. Chawla, N. Japkowicz, and A. Kotcz, “Special issue on learning from imbalanced data sets,” *ACM SIGKDD explorations newsletter*, vol. 6, no. 1, pp. 1–6, 2004.

[11] R. Longadge and S. Dongre, “Class imbalance problem in data mining review,” *arXiv preprint arXiv:1305.1707*, 2013.

[12] P. Liu, Y. Wang, L. Cai, and L. Zhang, “Classifying skewed data streams based on reusing data,” in *2010 International Conference on Computer Application and System Modeling (ICCASM 2010)*, vol. 4. IEEE, 2010, pp. V4–90.

[13] J. Van Hulse, T. M. Khoshgoftaar, and A. Napolitano, “Experimental perspectives on learning from imbalanced data,” in *Proceedings of the 24th international conference on Machine learning*, 2007, pp. 935–942.

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

[15] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, “Attention is all you need,” in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[16] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, “Improving language understanding by generative pre-training,” *URL https://s3-us-west-2. amazonaws. com/openai-assets/researchcovers/languageunsupervised/language understanding paper. pdf*, 2018.

[17] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman, “Glue: A multi-task benchmark and analysis platform for natural language understanding,” *arXiv preprint arXiv:1804.07461*, 2018.

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

[19] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[20] F. A. Gers, N. N. Schraudolph, and J. Schmidhuber, “Learning precise timing with lstm recurrent networks,” *Journal of machine learning research*, vol. 3, no. Aug, pp. 115–143, 2002.

[21] S. Hochreiter, Y. Bengio, P. Frasconi, J. Schmidhuber *et al.*, “Gradient flow in recurrent nets: the difficulty of learning long-term dependencies,” 2001.

[22] L. Zhang, W. Zhou, and L. Jiao, “Wavelet support vector machine,” *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 34, no. 1, pp. 34–39, 2004.

[23] C. S. Smith, “A class of complex modifiers in english,” *Language*, vol. 37, no. 3, pp. 342–365, 1961.

[24] X. Wu, S. Lv, L. Zang, J. Han, and S. Hu, “Conditional bert contextual augmentation,” in *International Conference on Computational Science*. Springer, 2019, pp. 84–95.

[25] O. Çelebi, “fasttext,” "facebookresearch/fastText/releases/tag/v0.2.0", 2019-03-18.

[26] S. Edunov, M. Ott, M. Auli, and D. Grangier, “Understanding back-translation at scale,” *arXiv preprint arXiv:1808.09381*, 2018.

[27] “Twitter dataset,” http://alt.qcri.org/semeval2017/, accessed: 2020-04-30.

[28] “Ag news dateset,” http://groups.di.unipi.it/~gulli/AG_corpus_of_news_articles.html, accessed: 2020-04-21.

[29] S. C. Wheeler, “Attributives and their modifiers,” *Noûs*, pp. 310–334, 1972.

Appendix A

Original results

Table A.1: The F1-score of Twitter dataset in SVM Classification

MODEL	The Percentage of Training SET				
SVM	20%	40%	60%	80%	100%
+ Ori-data	49.39	51.05	52.81	54.00	54.66
+ Over-sampling(SMOTH)	52.16	53.20	55.76	56.42	57.80
+ Down-sampling	51.06	52.21	52.11	53.09	53.38
+ Back-trans	54.30	55.70	56.35	56.69	56.87
+ Tuned BERT	53.05	54.84	56.25	57.51	58.38
+ BERT	53.55	54.76	55.74	57.23	58.19
+ fastText	52.92	53.87	55.14	56.49	56.69
+ Deletion	52.79	54.59	56.37	57.21	58.33

Table A.2: The F1-score of Twitter dataset in Bi-LSTM Classification

MODEL	The Percentage of Training SET				
Bi-LSTM	20%	40%	60%	80%	100%
+ Ori-data	35.13	39.56	43.66	45.36	47.22
+ Over-sampling(SMOTE)	43.04	46.10	47.69	50.15	51.16
+ Down-sampling	42.04	43.01	45.04	49.15	49.62
+ Back-trans	42.44	44.63	51.33	50.91	51.44
+ Tuned BERT	44.36	48.48	47.31	51.36	51.39
+ BERT	46.05	46.76	48.02	50.90	51.71
+ fastText	42.35	46.05	49.83	51.90	51.73
+ Deletion	43.14	46.39	47.22	50.23	51.25

Table A.3: The F1-score of Twitter dataset in fastText Classification

MODEL	The Percentage of Training SET				
fastText	20%	40%	60%	80%	100%
+ Ori - data	49.99	52.66	54.02	54.91	56.04
+ Over-sampling(SMOTE)	51.24	53.66	56.29	55.92	57.93
+ Down-sampling	50.24	52.69	55.15	55.19	56.45
+ Back-trans	54.13	54.62	56.21	57.43	56.12
+ Tuned BERT	52.45	54.35	55.14	56.37	56.61
+ BERT	52.68	52.72	54.14	55.35	54.55
+ fastText	52.62	53.66	55.34	55.78	55.27
+ Deletion	53.02	54.03	54.55	56.61	55.78

Table A.4: The Recall of Twitter dataset in SVM Classification

MODEL	The Percentage of Training SET				
SVM	20%	40%	60%	80%	100%
+ Ori - data	49.20	50.53	51.85	52.70	53.37
+ Over-sampling(SMOTE)	52.16	54.20	54.75	55.27	56.87
+ Down-sampling	52.16	54.20	54.75	55.27	56.87
+ Back-trans	55.42	56.46	57.01	56.82	56.56
+ Tuned BERT	53.04	54.85	56.80	58.00	59.22
+ BERT	53.68	54.90	56.29	57.86	58.89
+ fastText	54.63	56.34	58.45	60.13	60.88
+ Deletion	52.50	54.17	56.47	57.36	58.61

Table A.5: The Recall of Twitter dataset in Bi-LSTM Classification

MODEL	The Percentage of Training SET				
Bi-LSTM	20%	40%	60%	80%	100%
+ Ori - data	36.64	41.16	44.43	45.73	47.06
+ Over-sampling(SMOTE)	44.23	48.37	50.70	53.00	54.15
+ Down-sampling	42.35	44.74	46.97	51.10	52.79
+ Back-trans	47.74	45.97	51.15	50.66	52.62
+ Tuned BERT	46.35	48.38	50.84	53.27	55.86
+ BERT	46.09	48.37	51.70	51.52	53.15
+ fastText	45.35	47.30	51.95	50.10	52.49
+ Deletion	45.17	47.27	49.03	49.96	52.05

Table A.6: The Recall of Twitter dataset in fastText Classification

MODEL	The Percentage of Training SET				
fastText	20%	40%	60%	80%	100%
+ Ori - data	49.39	51.69	53.12	53.80	54.91
+ Over-sampling(SMOTE)	51.16	54.44	56.54	57.69	58.91
+ Down-sampling	50.61	52.81	53.93	55.25	56.12
+ Back-trans	54.13	54.44	58.04	58.84	58.01
+ Tuned BERT	52.67	54.38	56.91	57.86	58.46
+ BERT	52.52	53.47	55.93	56.32	56.96
+ fastText	52.03	53.14	55.33	55.78	55.87
+ Deletion	53.27	54.39	56.08	58.16	58.96

Table A.7: The F1-score of Ag News dataset in SVM Classification

MODEL	The Percentage of Training SET				
SVM	20%	40%	60%	80%	100%
+ Ori - data	84.58	86.16	86.87	87.72	88.03
+ Over-sampling(SMOTE)	86.42	87.37	87.65	88.38	88.50
+ Down-sampling	81.52	83.48	84.42	85.03	84.90
+ Back-trans	86.51	87.34	88.01	88.61	88.87
+ Tuned BERT	86.90	87.84	88.14	88.50	88.26
+ BERT	87.28	87.98	88.23	88.47	88.38
+ fastText	87.21	87.77	87.81	88.26	88.08
+ Deletion	85.16	86.33	87.49	87.74	88.15

Table A.8: The F1-score of Ag News dataset in Bi-LSTM Classification

MODEL	The Percentage of Training SET				
Bi-LSTM	20%	40%	60%	80%	100%
+ Ori - data	60.85	66.05	70.10	74.03	75.52
+ Over-sampling(SMOTE)	68.41	73.50	75.45	77.97	79.19
+ Down-sampling	31.82	46.82	56.49	63.87	69.32
+ Back-trans	68.13	76.81	75.69	75.69	79.17
+ Tuned BERT	70.90	74.73	76.28	77.75	79.37
+ BERT	69.09	74.04	76.41	78.39	80.67
+ fastText	70.18	73.50	76.42	78.56	79.10
+ Deletion	69.72	74.43	77.61	78.87	79.52

Table A.9: The F1-score of Twitter dataset in fastText Classification

MODEL	The Percentage of Training SET				
fastText	20%	40%	60%	80%	100%
+ Ori - data	49.99	52.66	54.02	54.91	56.04
+ Over-sampling(SMOTE)	51.24	53.66	56.29	55.92	57.93
+ Down-sampling	50.24	52.69	55.15	55.19	56.45
+ Back-trans	54.13	54.62	56.21	57.43	56.12
+ Tuned BERT	52.45	54.35	55.14	56.37	56.61
+ BERT	52.68	52.72	54.14	55.35	54.55
+ fastText	52.62	53.66	55.34	55.78	55.27
+ Deletion	53.02	54.03	54.55	56.61	55.78

Table A.10: The Recall of Ag News dataset in SVM Classification

MODEL	The Percentage of Training SET				
SVM	20%	40%	60%	80%	100%
+ Ori - data	49.20	50.53	51.85	52.70	53.37
+ Over-sampling(SMOTE)	52.16	54.20	54.75	55.27	56.87
+ Down-sampling	52.16	54.20	54.75	55.27	56.87
+ Back-trans	55.42	56.46	57.01	56.82	56.56
+ Tuned BERT	53.04	54.85	56.80	58.00	59.22
+ BERT	53.68	54.90	56.29	57.86	58.89
+ fastText	54.63	56.34	58.45	60.13	60.88
+ Deletion	52.50	54.17	56.47	57.36	58.61

Table A.11: The Recall of Twitter dataset in Bi-LSTM Classification

MODEL	The Percentage of Training SET				
Bi-LSTM	20%	40%	60%	80%	100%
+ Ori - data	36.64	41.16	44.43	45.73	47.06
+ Over-sampling(SMOTE)	44.23	48.37	50.70	53.00	54.15
+ Down-sampling	42.35	44.74	46.97	51.10	52.79
+ Back-trans	47.74	45.97	51.15	50.66	52.62
+ Tuned BERT	46.35	48.38	50.84	53.27	55.86
+ BERT	46.09	48.37	51.70	51.52	53.15
+ fastText	45.35	47.30	51.95	50.10	52.49
+ Deletion	45.17	47.27	49.03	49.96	52.05

Table A.12: The Recall of Twitter dataset in fastText Classification

MODEL	The Percentage of Training SET				
fastText	20%	40%	60%	80%	100%
+ Ori - data	49.39	51.69	53.12	53.80	54.91
+ Over-sampling(SMOTE)	51.16	54.44	56.54	57.69	58.91
+ Down-sampling	50.61	52.81	53.93	55.25	56.12
+ Back-trans	54.13	54.44	58.04	58.84	58.01
+ Tuned BERT	52.67	54.38	56.91	57.86	58.46
+ BERT	52.52	53.47	55.93	56.32	56.96
+ fastText	52.03	53.14	55.33	55.78	55.87
+ Deletion	53.27	54.39	56.08	58.16	58.96

TRITA -EECS-EX-2020:874