COMP90049 Project 1: Word Blending in Twitter

Anonymous

1 Introduction

People often use blend words when they post their opinions on Twitter. Plenty of Twitter users like using new words originating from blending two words that have already existed in the dictionary.

In this project, I have developed a method to detect blend words from a list of tokens. These tokens come from the tweets that users have sent. By using knowledge technology algorithms, the blend words can be detected effectively.

This paper is organized as follows. First, section 2 shows the literature review and section 3 presents the dataset that has been used in this project. Then, section 4 introduces the metrics that can be used to evaluate the effectiveness of this system. After that, section 5 presents the methodology of the system and the result of the experiment. Finally, section 6 summarizes this paper and presents the conclusions.

2 Literature Review

Some related studies have been done to use algorithms to detect blend words. For example, Cook Paul et al. have proposed an effective method to detect blend based on the existing machine learning algorithm [1]. Warintarawej Pattarapornhave compared between different ways of detecting blends and proposed a better algorithm of blend detection [2]. However, very few of them have used the N-Gram algorithm to identify blends.

3 Dataset

The Twitter data that have been used in this project are based on the data set presented in the paper of Jacob Eisenstein et al [3]. The dictionary is a list of about 370 thousand modified English words, which have been taken from the website https://github.com/dwyl/english-words. The true blend words are a list of tokens appearing in the original tweets that have been proved to be blend words [4-6]. This dataset is used to evaluate the effectiveness of the blends detecting algorithm.

4 Evaluation Metrics

In this paper, the following metrics will be used to evaluate the effectiveness of the blend detection system.

4.1 Accuracy

Accuracy is an important standard of system evaluation. It can be used to detect whether the system is useful to detect the blend words in the given dataset. It refers to the correct responses among the total number of words that have been used in the dataset. The formula of accuracy is as follows:

$$\text{accuracy}=\frac{tp+tn}{tp+tn+fp+fn}$$

where tp refers to the number of true positives, tn refers to the number of true negatives, fp refers to the number of false positives and fn refers to the number of false negatives [7].

4.2 Precision

Precision is another useful standard for system evaluation. It refers to the proportion of correctly predicted blends among the total number of predicted blends. The formula of precision is as follows:

$$\text{precision}=\frac{tp}{tp+fp}$$

where tp refers to the number of true positives and fp refers to the number of false positives [7].

4.3 Recall

Recall is also a meaning standard to evaluate the blend detection system. It refers to the ratio of correctly predicted blends among total number of true blend words. The formula of recall is as follows:

$$\text{recall} = \frac{tp}{tp + fn}$$

where tp refers to the number of true positives and fn refers to the number of false negatives [7].

5 Methodology and Result

In this project, I use the N-Gram algorithm to calculate the similarities between different words. The method of using N-Gram and the result of detecting blends is shown in this section.

5.1 Introduction to N-Gram

N-Gram is a good way to find the similarities between different words. By using N-Gram algorithm, we can find a series of substrings of a given word. If most of the substrings of a given word are the same as that of the other word, it means these two words are quite similar.

In Python, there is a package called 'ngram'. It can be used to compare the similarities between different words, based on the basic N-Gram algorithm. The function of comparing different words is 'ngram.compare'. By using this function, we can get the exact similarities of two words. The similarity between two words can be used to detect whether a word is a blend, because blend words are normally similar to their components. To increase the precision of detecting blends, we need to set a value of similarity. If the similarity between the candidate word and one word in dictionary exceeds the value we have set, we then believe that the candidate word is a blend. Therefore, it is important to choose a suitable value so that the blends can be detected precisely.

5.2 Methodology

In this project, Python is used to try to detect the blend words. Specifically, I have used 'ngram.compare' to calculate the similarity between two words, which helps the system detect the blend words. I have tried to set the critical similarity value of the blend detection system, which is the threshold for blend words detection. Apart from that, I have chosen N=2 for the N-Gram algorithm that is used to detect blends so that most of the possible blend words are likely to be detected.

The most important part of the source code used for blends detection is shown below:

```
        for j in temp_:
            sim = NGram.compare(self.i, j, N=2)
            if sim >= threshold:
                flag=True
                break
        if flag==False:
            if self.i in ans:
                fn = fn + 1
            else:
                tn = tn + 1
            threadlock.acquire()
            print(str(self.loc) + ":" + self.i + " is not a blend!")
            threadlock.release()
        else:
            if self.i in ans:
                tp = tp + 1
            else:
                fp = fp + 1
            threadlock.acquire()
            print(str(self.loc) + ":" + self.i + " is a blend!")
            threadlock.release()
```

To test how the blend detection system performs, I have set the threshold as 0.5 and run the blend detection system. The result of the experiment has shown that the system performs quite well if the value is 0.5.

5.3 Result

When the threshold value is set as 0.5, I run the blend detection system to try to detect the blend words in the candidate lists. When the system runs, it automatically compares the word in the candidate lists and words in the dictionary dataset. Based on the result of N-Gram similarities, the system can predict whether a given word is a blend or not. The accuracy, precision and recall of the system are shown in the table below:

Evaluation Metrics	Values
Accuracy	22.87%
Precision	1.02%
Recall	88.08%

Table 1- System Evaluation Table

The data in this table shows the value of accuracy, precision and recall of this system. It is evident that the value of system recall is highest and that of system precision is incredibly low. This is because the number of false positives is extremely high. Some candidate words are quite similar to words in the dictionary, but they are not true blend words. For example, 'abot' is similar to 'about' but it is not a blend. Therefore, the system is effective to some extent, but it still needs to be improved.

6 Conclusions

This report proposes a system that can predict the blend words by using N-Gram algorithm. The result of system evaluation indicates that the system is useful to predict blends to some extent. However, the precision value of the system is very low and we need to alter the algorithm to try to further enhance the system precision of predicting blends.

References

[1] Cook, P., & Stevenson, S. (2010). Automatically Identifying the Source Words of Lexical Blends in English. In *Computational Linguistics*, Volume 36(1), pages 129–149.

[2] Warintarawej, Pattaraporn. (2013) Automatic Analysis of Blend Words. *EBSCOhost*.

[3] Jacob Eisenstein, Brendan O'Connor, Noah A. Smith,and Eric P. Xing. (2010) A latent variable model forgeographic lexical variation. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing (EMNLP 2010)*, pages 1277–1287.

[4] Deri, A. and Knight, K. (2015) How to Make a Frenemy: Multitape FSTs for Portmanteau Generation. In *Human Language Technologies: The 2015 Annual Conference of the North American Chapter of the ACL*, pages 206–210.

[5] Das, K. and Ghosh, S. (2017) Neuramanteau: A Neural Network Ensemble Model for Lexical Blends. In *Proceedings of the The 8th International Joint Conference on Natural Language Processing*, pages 576–583.

[6] Cook, P. and Stevenson, S. (2010) Automatically Identifying the Source Words of Lexical Blends in English. In *Computational Linguistics*, Volume 36(1)

[7] Powers, David M W (2011). Evaluation: From Precision, Recall and F-Measure to ROC, Informedness, Markedness & Correlation. *Journal of Machine Learning Technologies*. Volume 2 (1), pages 37–63.