COMP90049 Knowledge Technologies
Assignment 1: Word Blending in Twitter Report

1 Introduction

Lexical blends — words such as brunch (breakfast + lunch) and belieber (believer + bieber) — are popular nowadays. In this report, we implement a source words recognization system to detect blends among frequent terms and detect the components of a given blend based on approximate string searching and matching method.

2 Dataset

In this report, a dictionary[1] is used for searching for possible components. A twitter datasets with 62318 tweets (Eisenstein et al., 2010) and another with 364369 tweets (from a Kaggle Prediction Competetion[2]) are involved for calculating the frequency of words appearing on Twitter. A candidate dateset and a blend dataset (Deri and Knight, 2015; Das and Ghosh, 2017; Cook and Stevenson, 2010) are involved for the purpose of evaluating the method.

3 Methodology

The core idea of the method is to find the most likely components of a given term. For detecting whether a term is a lexical blend, we use the similarity between the term and a blend recoined by its most likely components. For detecting components of a given lexical blend, we retrieve the most likely components based on our ranking method.

3.1 Word Frequency

According to the study of Lehrer (2003), people can easily recognize the source words of a given blend if the source words are common in life. This means the term frequency can be a useful feature for defining a blend's components. To caltulate the term frequency for each term in the dictionary dataset and the candidate dataset, we count the number of times they appeared on the the preprocessed tweets dataset.

3.2 Candidate Set

Most lexical blends are words such as brunch (breakfast + lunch) that are composed of a prefix of one source word and a suffix of another. Based on this characteristic, for a given word, we first generates each possible prefix-suffix pairs of it. And then we finds words with specific prefixes and suffixes in the dictionary. To reduce the search time, we have established a set of index tables for all prefixes and suffixes in the dictionary in advance. After that, we generates a list of candidate word pairs by taking the Cartesian product of the prefix words and suffix words. This method is mentioned by Cook and Stevenson (2010). To increase its detection accuracy and reduce its complexity, an improved version is presented here.

3.2.1 Prefix-Suffix Pairs

To generate a candidate set with high probability, we restrict the length of the prefix and suffix to be two or more. And according to the discovery from Gries (2004) and Kelly (1998), the proportion of the former source word in a blend is less than that of the latter. So, in a candidate prefix-suffix pair, the length of the prefix will not be longer than that of the suffix. Besides, for a blend with 4 letters, we take the first two letters of it as its prefix and the last three letters of it as its suffix. This method is based on the pronunciation feature of blend and knowledge of vowels.

3.2.2 Cartesian product

For each prefix-suffix pairs, we find a list of words which contain the prefix and a list of words which contain the suffix. Then, a list of candidate pairs is generated by taking the Cartesian product of the prefix words and the suffix words. The candidate set is comprised of candidate pairs from all of the possible prefix-

[1] A slightly-altered version of the data from https://github.com/dwyl/english-words

[2] https://www.kaggle.com/c/whodunnit

suffix pairs created by the section 3.2.1. For those duplicate pairs, we count the number of times it repeats, but only keep one in our candidate pairs. For example, the candidate pair *bruke mispunch* appears twice: once from the prefix-suffix pair *br unch*, once from the *bru uch*, and the count of it would be two. In addition, any prefix words or suffix words with zero word frequency would not be added to the candidate pairs set.

3.3 Approximate Matching

3.3.1 N-Gram

N-Gram is a method for finding the best approximate string match. It is potentially useful for finding approximate match prefix and suffix. Through testing, we find out that not taking the ordering of the word into account can provide a better precision. As a result, we use Unigram Models as our system's approximate string matching method.

3.3.2 Blend Inverse Restoration

To obtain an object that can be compared with a given term, we propose a new method, which we call it Blend Inverse Restoration. For a pair of prefix source and suffix source, we first use the prefix source and the suffix to generate a Suffix-Restore Blend. We combine the suffix with the substring before the first vowel letter of the prefix source (if the first letter in the suffix is a vowel letter, the first vowel letter of the prefix source would be included in the substring, otherwise not). Secondly, we use the prefix and the suffix source to generate a Prefix-Restore Blend. We combine the prefix with the substring after the first vowel letter of the suffix source (if the last letter in the prefix is a vowel letter, the first vowel letter of the suffix source would not be included, otherwise not). For example, for a candidate pair *dumping(dump), protester(ster)* of *dumpster*, the Suffix-Restore Blend of it is *duster*, and the Prefix-Restore Blend of it is *dumper*.

3.3.3 Distance Calculation

Using the Blend Inverse Restoration method from the Section 3.3.2, for each candidate pair, we calculate a distance for it with Formula 1:

$$Distance = w \times D_1 + (1 - w) \times D_2 \qquad (1)$$

In formula 1, D1 is the N-Gram Distance between the term and the Suffix-Restore Blend, D2 is the N-Gram Distance between the term and the Prefix-Restore Blend and w is a weight chosen between 0 to 1.

3.4 Blend Detection

For a given term, we consider it to be a blend if the distance (based on section 3.3.3) between the most likely candidate pair and itself is less than 10 (Optimal choice obtained by tests).

3.5 Components Detection

For a given lexical blend, we retrieve the most likely candidate pair based on the distance (the one with the smallest distances).

4 Analysis

4.1 Complexity

The most time-consuming part of our method is to calculate the real distance for each candidate pair. This is because the size of the candidate set is huge. And we find out that reasonable reductions of candidate sets can not only improve its efficiency but also improve the accuracy and precision. For example, by specifying that members entering the candidate set must appear at least once on the tweets dataset, the precision of Component Detction improve 3%. The complexity of using our method to find the most likely components of a given term is:

$$O(n^2)$$

4.2 Evaluation

4.2.1 Blend Detection Evaluation

For the Blend Detection part, we consider the accuracy, precision, recall and F1 Score of it. The F1 Score is the harmonic mean of recall and precision. The formulas of them are list below:

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

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$F1\ Score = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

In the Blends Detection method, a term appears less than N times on the tweets dataset would not be considered as a blend. Result of different N is show in Table 2. With the increase of N, the precision, accuracy, and F1 Score increase. This is because a blend is coined when people use it frequently. The recall decrease because the increase of the threshold prevent some true blend from being taken into account. Besides,

our method can't handle the blends which are not formed by combining a prefix of one source word with a suffix of another source word. The precision is low because for a given term, it is easy to find two source words from a given prefix and suffix pair.

Metric	N = 1	N = 10	N = 20	N = 50
Precision	0.0159	0.0409	0.0477	0.0672
Recall	0.8742	0.3642	0.2450	0.1391
Accuracy	0.5101	0.9160	0.9488	0.9747
F1 Score	0.0313	0.0728	0.0798	0.0905

Table 1: The Result of Blends Detection

4.3 Components Detection Evaluation

For the Components Detection part, we only consider the precision of it because the false-negative result is hard to define. Precision of different w is show in Table 2. The precision of detecting the prefix source is low because the proportion of the former source word in a blend is less than that of the latter. For example, *breakfast* only takes up the first two letters of *brunch*, while almost all letters of *lunch* participate in the formation of the blend. With so little information, it is almost impossible to find the source in a huge candidate set. To achieve a better precision, We can extract information from its context on tweets dataset. But in this way we need a new assessment method.

w	P(prefix source)	P(suffix source)
0.0	0.0874	0.2896
0.25	0.0874	0.2951
0.5	0.0874	0.3005
0.75	0.0929	0.3005
1.0	0.1256	0.2513

Table 2: The Result of Components Detection

5 Conclusions

In this report, we propose a method to find the most likely components of a given term, which can be used to detect blends or detect the components. For Blends Detection, the accuracy of the method is high, but the precision is relatively low. For Components Detection, the method can obtain a high precision of predicting suffix sources but a low precision of predicting prefix sources. There are still plenty of ways to improve our method.

References

P. Cook and S. Stevenson. Automatically identifying the source words of lexical blends in English. *Computational Linguistics*, 36(1): 129–149, 2010.

K. Das and S. Ghosh. 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. Taipei, Taiwan, 2017.

A. Deri and K. Knight. 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. Denver, USA, 2015.

J. Eisenstein, B. O'Connor, N. A. Smith, and E. P. Xing. A latent variable model for geographic lexical variation. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing (EMNLP 2010)*, pages 1277–1287. Cambridge, USA, 2010.

S. T. Gries. Shouldn't it be breakfunch? a quantitative analysis of blend structure in english. *Linguistics*, pages 639–668, 2004.

M. H. Kelly. To "brunch" or to "brench": Some aspects of blend structure, 1998.

A. Lehrer. Understanding trendy neologisms. *Italian Journal of Linguistics*, 15:369–382, 2003.