Detect Lexical Blends by Similarity of Spells and Meanings

Anonymous

1 Introduction

Blend words are words that have been coined by the fusion of parts of two different words. Given two words to make a blend word, only several options exist at most. On the other hand, given one blend word to parse into two words, a numerous number of combinations exist. How can we detect the blend word in an efficient way? The aim of this report is to implement a system that detects blended word. The initial system was implemented based on one of the approximate string matching to detect blend words. An attempt to prevent unrelated combination of source words by combining the method of calculation of word vectors was further made.

2 Dataset

The data (candidates.txt) involves candidates for blend words extracted from the twitter data set(Eisenstein et al., 2010)[1]. The dataset comprises of 16684 word tokens which do not appear in the dictionary. The list of blend words(blends.txt), which were manually identified as being lexical blends using resources in [2][3][4](Deri and Knight, 2015; Das and Ghosh, 2017; Cook and Stevenson, 2010)) was also provided. Another dataset involves a word dictionary, which contains approximately 370K English words from online resources[1], from which we get help in order to detect blend words.

3 Evaluation Metrics

To evaluate the implemented system, the following terms will be used in this paper. All the evaluated result in this paper was tested on a full data set.

- Accuracy: Accuracy was calculated by counting number of the correctly returned results and number of data not returned by the system which is not listed in the blends.txt divided the number of the full data set.
- Precision: Precision was calculated by counting the correctly returned result which is in the blends.txt out of the full data set.

4 Similarity in spells

4.1 Methodology

Among approximate string matching methods, Jaro-Winkler similarity was found in that blend words have remarkable patterns of having prefix and suffix. Instead of simply counting the matching characters, this method returns the point how similar two given words are, weighted by the matching prefix. In this paper, the Jaro-Winkler Similarity was used to calculate similarity between candidate blend words and a token in a dictionary and the returned result with the high scores are likely to match.

4.2 Analysis

After an analysis of the provided blends.txt file was conducted, some patterns between the blend words and source words were found. In most cases, the prefix of blend words represent the first source word and the suffix of blend words are for the second source word. However, there are cases where the prefix of the second source words form the suffix of the blend words. Based on this simple rule, the system was initially implemented. Most importantly, simply setting a threshold for one affix matching and counting two was never enough. From calculation of some samples, sum of Jaro-Winkler similarity of two source words were in a certain range approximately from 1.3 to 1.9 as seen in the Table 1. Therefore, if the sum is below this range, it is not likely to be the blend word, and if it is over the range, it is likely to be exact matching rather than a blend word.

[1] https://github.com/dwyl/english-words

	buppie	funemployment	macbook
word1	black	fun	macintosh
sim	0.49	0.8	0.74
word2	yuppie	employment	notebook
sim	0.89	0.92	0.61
SUM	1.38	1.72	1.35

Table 1: Examples of words calculated by Jaro-Winkler similarity

4.3 Affixing blend words

After an analysis of the provided blends.txt file was made, two patterns were considered in this paper: 1) the prefix of a blend word match the the prefix of the first source word and the suffix of blend words match the suffix of the second source word. 2) the prefix of a blend word match the prefix of the first source word and the suffix of the blend words match the prefix of the second source words. However, the second pattern was not considered in implementation of the word vector similarity system, which will be explained later, due to the limited resources.

4.4 Scaling Parameter Variation

In Jaro-Winkler similarity, scaling parameter is used for weighting the same prefix. In an experiment, less than 0.99 performed well. Adjustment was made between 0.7 and 0.9 to perform the best. In terms of comparison with the suffix of the blend words, the same formula was applied after reversing the characters of the suffix of the blend word.

4.5 Implementation

Most of the code was written by myself except Jaro-Winkler library and Gensim library. I implemented Jaro Winkler into the code but used the library instead in order to reduce time that is taken in finding transposition in my algorithm.

- The prefix of the candidate which passed the threshold of Jaro-Winkler similarity, indexed to (i)th position of the blend word will be stored in (i)th array.
- The suffix of the candidate which passed the threshold of Jaro-Winkler similarity, indexed from (i+1)th position will be compared to (i)th prefix of the candidate and update the highest sum of Jaro-Winkler similarity in (i)th array.
- When finishing checking one blend word, iterating the array that has index as the position of the blend word characters, find the highest sum of Jaro-Winkler similarity.

4.6 Results

The results tested against the full data set are shown below.

Accuracy	0.433
Precision	0.013
Recall	0.803
Time/s	3932

Table 2: Approximated Efficiency

4.7 Analysis

As seen in recall, it is able to say that an attempt to find the blend words from the candidate list which are in the given correct blend list by using Jaro-Winkler similarity was successful. However, it generated candidate lists far more than enough, thus the accuracy and precision is low. There is another problem that the predicted source words of the blend words barely make sense put together.

5 Semantic Similarity

Considering the problem of low accuracy and the irrelevance of the source words, the mechanism for finding more plausible relationship between two source words was needed. Inspired that similarity between words can be calculated by cosine similarity of two word vectors[5], an attempt to get semantic similarity between words was made in order to prevent prediction of meaningless combination of words.

5.1 Methodology

In terms of calculation of word vectors, Gensim package, among which "glove-twitter-25" was used, as it will represent the relationship between words better given that the data set is extracted from twitter. In a test with some words, it returned a number well representing the relationship between two words as seen in table. From the test, I concluded that 0.5 is a threshold enough to comprise a blend word. Note that many codes were removed to decrease complexity due to long processing time. Details to note about the modified implementation of the system are:

- Jaro-Winkler similarity is now used only for threshold whereas its total sum was critical key to determining blend words in the previous system.
- The suffix of the candidate which passed the threshold of Jaro-Winkler similarity indexed from (i+1)th position will be compared to prefix of the candidate indexed to (i)th position, which also passed the threshold of Jaro-Winkler similarity, to calculate word vector similarity.
- Comparing the best value per position per word, the best value that pass both Jaro-Winkler and word vector similarity will be returned.

The results of applying this methodology can be seen in Table 3.

Accuracy	0.766
Precision	0.016
Recall	0.585
Time	15685

Table 3: Results of combined Jaro-Winkler and Word vector similarity

5.2 Analysis

This algorithm, which was adding word vector similarity to initial system, was successful, achieving higher accuracy and low precision. Most importantly, the source words that comprise the blend words now makes more sense than the initial system as seen in Table 4. Due to its long processing time, many codes from the Jaro-Winkler version was removed, so recall was not as good as before. However, if word vector function was added without shrinking the previous system, it is expected to achieve greatly higher in all aspects of evaluation metrics.

Blend	Word1	Word2
blaxican	blasting	mexican
camcorder	camshaft	recorder
wikipedia	wiki	logopedia

Table 4: Examples of the predicted source words in the modified model

6 Conclusions

Overall, through careful analysis of the original data along with dividing positions for prefix and suffix accordingly and adjustment of parameter, acceptable result was achieved. Not only finding out the blend words, it was possible to get the meaningful a pair of source words. Although it was implemented in a reduced scale algorithm due to the limited computing power and time, high performance was achieved by the two methods combined above. However, in this experiment, only similarity in spells and meaning was considered. For example, phonetics should be considered here as people tend to make blend words easy to pronounce as one word.

References

[1] Jacob Eisenstein, Brendan OConnor, 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 12771287

[2]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 206210

[3]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 576583

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

[5]Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig.2013c. Linguistic regularities in continuous space word representations. In Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. Association for Computational Linguistics, pages 746751. http://aclweb.org/anthology/N13-1090.