# COMP90049 Project 1 Report: Word Blending

**Anonymous**

## 1 Introduction

This report aims to implement Local Edit Distance algorithm (herein LED) and Jaro-Winkler Similarity algorithm (herein JWS) to detect blending word, in which the word candidates comes from a data set comprising frequent terms in Twitter data. LED algorithm calculates the approximate string match distance between two strings. JWS algorithm aims to approximate the similarity between the prefix and suffix of two strings. LED and JWS are used to decide whether the prefix and the suffix of a word are close enough to a word that is already in the English Dictionary. Therefore, the word can be marked as a blending word if the prefix and suffix of it both are similar closely with any word in the English Dictionary by using LED and JWS.

## 2 Dataset

In the experiment, three datasets are used: Candidates, Dict and Blend. Candidates dataset comprises 16925 tokens from tweet text (Eisenstein et al., 2010) of Twitter, which are not included in the English Dictionary. Dict dataset lists approximately 370K English entries from the English dictionary. Blend dataset(Deri and Knight, 2015; Das and Ghosh, 2017; Cook and Stevenson, 2010) lists the tokens which appear in the tweets and which are identified manually as lexical blends.

## 3 Evaluation Metrics

There are two terms used to evaluate the methodologies in this paper.

### 3.1 Precision

The term Precision describes the fraction of the correct blend words among all predicted blend words. Precision shows the accuracy of the blend detection method.

### 3.2 Recall

This term Recall indicates the proportion of correct blend words among all true blending words which are manually identified and recorded in the dataset Blend. It describes whether the method is good enough to detect all true blending words. In dataset Blend, there are 183 identified blending words. However, only 151 of 183 blending words are included in the dataset Candidates, which means the method can detect no more than 151 true blending words.

## 4 Methodology

This section introduces the technology and methods used in this paper.

### 4.1 Dataset Preprocessing

Before the detection of blending words in this research, proper dataset preprocessing is needed to filter the initial dataset Candidates and store the tokens more effectively and efficiently to save time and obtain a better outcome. In this paper, an ordered data tree structure Trie is used, which is a kind of search tree also known as the prefix tree. In the tree of Trie, every descendant of a node has a common prefix of the string associating with the node. Therefore, Trie helps save time when the program is trying to find some string with a common prefix. In terms of suffix, Trie also can be used with the reverse of the suffix and the string.

### 4.2 Local Edit Distance

Local Edit Distance (LED) calculates the minimum operation used to convert a string to another string, which indicates the similarity between two strings. Possible operations between strings consist of Insertion, Deletion, Replace and Match. The main aim of LED is to find the best substring out of one string. In this paper, this method can be used to determine whether one prefix or a suffix is the best substring of one word.

### 4.3 Jaro - Winkler Similarity

Jaro - Winkler Similarity is another useful method fo detecting words with similar prefix or suffix. In the extension of Jaro Winkler Similarity, more weight is given to strings with matching prefixes. This method helps detect and determine whether a token from Candidates has the same prefix or suffix as one word from dataset Dict.

## 5 Blending Words Detection

### 5.1 Basic Design

During the process of detection, every token from Candidates will be divided into the front part and the latter part. With the use of Local Edit Distance, the front part will be compared with the front part from every word in Dict. If the distance between these two front part is greater than the half length of the front part of the candidate, the candidate will be determined to be of high possibility to be a blending word. However, LED focus on finding the same substring rather than same prefix between two strings. Therefore, Jaro - Winkler Similarity will be introduced here. JWS focus on detecting whether two string have the same prefix. JWS is used to calculate the similarity after the candidate meets the threshold of LED. If the similarity achieves 0.85, then the candidate is considered to be of high possibility to have the same prefix as the word in Dict has. As for the latter part of tokens from Candidate, the letter order of the tokens and English words are reversed and then the same detecting process described above can be used here to calculate the distance and the similarity. A token can be considered as a blending word only if the front part and the latter part of the token meet the threshold of LED and JWS through the calculation of the process above.

### 5.2 Assumption

1. All English words in dict.txt are correct.
2. All blending words in candidates are in blends.txt.

### 5.3 Initial Value of Parameters

In the calculation of LED, the initial value of parameters (m, i, d, r) is set to be (1, -1, -1, -1), and the threshold of LED is set as half length of the front part or the latter part of the candidate. In the calculation of JWS, the threshold is 0.85 and the scaling value is 0.1.

### 5.4 Result

The result is showed in Table 1.

| | |
|---|---|
| Precision | 0.0125 |
| Recall | 0.9140 |

Table 1: The result of the Initial Blending Detection

### 5.5 Analysis

It shows low precision and high recall in the initial blending detection. Low precision indicates that the number of tokens marked as blending word is much more than the actual number of blending words. Besides, some blending word, which blends short prefix and long suffix or vice versa, failed to be recognized in the result because the length of the front part and the latter part of a token can only be the half length of the token at most. Table 2 shows some example of missed blending word.

| Blends | Component 1 | Component 2 |
|---|---|---|
| daycation | day | vacation |
| camcorder | camera | recorder |
| irregardless | irrspective | regardless |

Table 2: Some Missed Blends Examples

## 6 Modification to the Blending Detection

To increase the precision of the detection, it is feasible to eliminate the tokens marked as blending. In the dataset Candidate, there are lots of nonsense tokens with more than 3 repeated letters in one token and there are few words with more than 3 repeated letters in the English dictionary generally. Therefore, these useless words can be filtered and excluded before the detection. To exclude some tokens that are not blending words, the length of the front and the latter part is changed from half length of the token to the half length of the token plus 1 to make a one-letter overlap. In the calculation of LED, the threshold value is also changed from half length of the token to the 0.85 * length of the token.

### 6.1 Customised Parameters

In the calculation of LED, the value of parameters (m, i, d, r) is set to be (1, -1, -1, -1), and the threshold of LED is changed to be 0.85 * (length of the front part or latter part of the candidate). In the calculation of JWS, the threshold is 0.85 and the scaling value is 0.1.

## 6.2 Result after Modification

The result is shown in Table 3:

| | |
|---|---|
| Precision | 0.0198 |
| Recall | 0.2980 |

Table 3: The result of the Blending Detection with Modification

## 6.3 Analysis

The precision increases slightly from 0.0125 to 0.0198 whereas the recall drops sharply from 0.9140 to 0.2980. This is because the preprocessing and the modification of parameter value limits the number of candidates marked as blending. However, in the meanwhile, it excludes some real blending tokens.

## 7 Conclusions

In conclusion, the preprocessing of the dataset is important and LED and JWS can be used to detect blending words with high recall and low precision. In the modification of the detection method, it should be noticed that the real blending tokens in Candidates should not be excluded when the detection try to avoid mark irrelevant tokens as blending.

## 8 Summary of Relevant Literature

Researcher Warintarawej and Pattaraporn developed top-k classification to detect French blend words (Cook and Stevenson, 2010). In the detection, they consider three different features of candidate words: morpho-phonological stems, character N-grams and syllables. Moreover, they developed a novel method named Enqualitum to identify blending words automatically.

## 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.