COMP90049 Project 1 Report: Word Blending in Twitter

Abstract—**Twitter Blends words are words of similar meaning combined together to form a single word. This process of blending is often done to the word limit imposed on making a tweet in a twitter feed. These words have retained their meaning and have become colloquial around the globe. However these words are not part of the original dictionary hence maintaining a record of all these word blends becomes difficult. Hence A system working towards twitter blend identification has been developed and analysed in this paper.**

***Index Terms*— hamming distance, n-gram similarity**

Introduction

A TWITTER blend word consists of a word formed as a result of amalgamation of two component words. These component words can either be part of the english dictionary or the modern dictionary wherein proper nouns like places also have their corresponding entries. In order to build a system that can infer whether a word is a blend word or not it is important to identify corresponding component words from which the blend word originates. A key feature to take into account is these component words that would be identified as the origin words of a given blend word may or may not be the actual words from which the blend word originates. This paper addresses the system developed whose goal is to identify if a word is a blend word or not. Identifying the actual origin words forming these blend words is not in the scope of this system. The system develops a method of blend word identification and derives inferences from the system. Having studied the proposed system, as of this paper it has been updated to better identify if a word is a blend word or not.

Literature Review

Cook, P., & Stevenson, S. (2010), attempted to perform lexical blend identification by using the Twitter API and regular expressions. The idea behind this approach is identifying all words or terms that are new to an author or have been coined by the author of a tweet for the first time. This data is obtained by passing 27 different types of regular expressions as parameters to the Twitter API that gives about 10% of tweet related information[4]. Based on this system is trained and successfully identifies 57% of words as tweet blend in a given set of candidates. Grieve, J., Nini, A., & Guo, D. (2018), made use of lexical innovation and mapping it to identify blend words and identifying the corresponding component words. In the system developed, a corpus of words across various tweets is gathered and analysed over a timeline to identify if the use of a certain word has increased significantly over the timeline[1]. All such words are recorded and compared with a dictionary file to conclude if they are blend word or not. Both papers tackle the problem of lexical blend identification using Machine Learning techniques. In this paper an attempt is made to identify the word blends with similarity and distance methodologies.

Data Input and Preprocessing

A. *Preliminary Input*

The system takes a set of words which need to be identified as blend words as input. This set of words is provided in a text file namely candidate.txt file. In order to identify possible component word(origin words) a dictionary file in text format is provided. To ensure correctness of the system's identification process a list of Twitter tweets are clustered into a text file. These tweets may or may not contain a blend word in them. Initially this tweet.txt file is taken and each tweet is passed to a method to pick out all the words that are not part of a dictionary. These words are pushed into a list iteratively. The assumption is words that are not part of the dictionary are blend words.

B. *Final Input*

After the above mentioned pre-processing of the twitter tweets the list of blend words is pushed into a text file blends.txt. This file contains all possible blend words and their component words. This file does not act as a golden standard for the system but is merely used as a comparator file once the program has identified its own list of blend words. For all intentions the blends.txt file acts as test cases whose labels we are aware of. The final inputs to the system are: A candidates file with words who need to be identified as whether they are blend words or not, a dictionary to identify actual words, a blends file to check the correctness of the system's output.

Methodologies

The system uses two different algorithms to implement blend identification. The following are the methods used and its corresponding functioning:

C. *Hamming Distance*

Hamming distance is distance measure to identify how different a given word is from another. Let each word be a vector, then hamming distance is equal to the summation of all the indices where vector Wi is different from Wj. Mathematically it is represented as:

$u, v \in Fn$

$d(u, v) = v - u$ [5]

For example consider the words hold and hole.

The following is the index vise calculation of hamming distance

Index	**0**	**1**	**2**	**3**
hold	h	o	l	**d**
hole	h	o	l	**e**

Table 1.1

In the above table we conclude that the words differ at index 3 or the distance difference is one, since only one index is a mismatch. Some hamming distance approaches allocated weights to each index, in that case the distance may vary on not just the mismatch but also 'how far apart' the mismatches are.

D. *n-gram similarity*

The idea behind the working of the n-gram method is finding longest common subsequence in an n-gram inclusion. Mathematically,

$sn(\Gamma k,l) = 0$ if $(k = n \wedge l < n) \vee (k < n \wedge l = n)$

Let $\Gamma ni,j = (xi+1 \ . \ . \ . \ xi+n, yj+1 \ . \ . \ . \ yj+n)$ be a pair of n-grams in X and Y[7]

N-gram in simple terms identify the similarity between two given strings by index based matching of $i = j$ and iterate this process till $i = j+n$, where n is the number of grams.

For example consider the words cane and crane. The index matching is as follows:

Index	**0**	**1**	**2**	**3**	**4**
cane	c	a	n	e	-
crane	c	r	a	n	e

Table 1.2

As observed there is index mismatch after the first index. However the strings can be claimed to be similar considering they match at an index difference of +1. That is above words are similar according to unigram similarity.

Proposed System

A. *Business Logic*

The system aims to perform word blend identification. Initially the words to be identified are taken into a list. Each of these words are divided as substrings prefix and suffix. This division is based on predefined threshold value. For the purpose of this system, this threshold is half of the word length. After having acquired a prefix and suffix value, the system searches the dictionary file non-exhaustively for the words whose prefix matches with the prefix acquired and words whose suffix matches with the suffix acquired. The first occurance of words matching the prefix and suffix respectively are identified as the component words. Since component words are identified the word is declared as a blend word.

B. *System Implementation*

The system is developed using Python as the main programming language. An input candidate file of about 16684 words is provided. The dictionary file has about 370059 entries. To get words with a given prefix and suffix in the dictionary file, built in methods startswith() and endswith() are used. These methods are developed in accordance to the Hamming distance of identifying similarities between two words or strings. That is each of the above mentioned methods makes an index based comparison and a boolean value of true is returned should all indices match, else it returns false. In our proposed system, if we can obtain two words which each satisfy either the startswith() or endswith() condition, these words are the component words of the given word and hence the given word is a blend word.

Comparative analysis

The output of the above system is compared with the blends file to calculate precision(pr), accuracy(ay), recall(r) and F1 score(F1). The values are as follows:

Parameters	**pr**	**ay**	**r**	**F1**
Values	0.0052	0.0052	1.0	0.010

Table 2.1

Precision is a measure of the ratio of corrected answers to that of attempted answers. Accuracy is a measure of the ratio of correct answers to total words of input. Recall calculates correct answers among total words that are truly blends. F1 score is the harmonic mean of recall and precision. As observed in the table above, the precision of the system is significantly low. This is because the system is able to identify component words for all the words of candidates from the dictionary. Best example to explain the above statement is the word dinner. Dinner is a proper word present in the english dictionary, however, when passed as an input in the system it takes the prefix 'din' and 'ner' as the suffix. From the dictionary, it is able to fetch **din**e as one component word and in**ner** as another. Thus every value in the candidate file will find its corresponding component words in the dictionary irrespective of whether they are the true origin words. Hence when a comparative analysis between the blends file and output of our system is identified, the count of true positive is very low leading to a small value of accuracy and precision. It is important to note that the values of accuracy and precision is the same. This is because the system manages to attempt and provide conclusive results to all of its input words. Hence there is no distinction between total number of words and attempted answers. The recall gives a measure of 1 because all correct answers are present in the blend file. Hence recall is not the best way to identify whether this system is good identifying word blends or not.

Updated System

From the calculated precision it is clear that the proposed system is not efficient at identifying twitter blend words. Hence the system is updated to use n-gram similarity.

A. Business Logic

This system is an improvisation of the proposed system. Initially the words to be identified are taken into a list. Each of these words are divided as substrings prefix and suffix. This division is based on predefined threshold value. For the purpose of this system, this threshold is half of the word length. After having acquired a prefix and suffix value, the system searches the dictionary file non-exhaustively for the words whose prefix matches with the prefix acquired and words whose suffix matches with the suffix acquired. The first occurance of words matching the prefix and suffix respectively are identified as the component words. This system further incorporates n-gram to identify words which have index variation of 'n' from the prefix or suffix acquired. The words that are most similar to the input word are recorded. These words are identified as components of the word and the input is declared as a blend word.

B. System Implementation

The updated system is also implemented using Python. It makes use of the package similarity, from which Ngram module is imported. The system uses 2 indices variation for similarity calculation that is it uses Ngram = 2

Comparative analysis

The output of the above system is compared with the blends file to calculate precision(pr), accuracy(ay), recall(r) and F1 score(F1). The values are as follows:

Parameters	pr	ay	r	F1
Values	0.31	0.31	0.42	0.365

Table 2.2

From the above table it is observed that precision and accuracy values are still the same, since the system can still conclusively determine whether a given word is a blend or not.

Proposed system and Updated System

From tables 2.1 and 2.2 it is clear that the updated system is an improvement on the originally proposed system. An increment is observed in precision indicating that the updated system is better at identifying the right word blend. Change in recall and F1 score leads to understanding the nature of false positives.

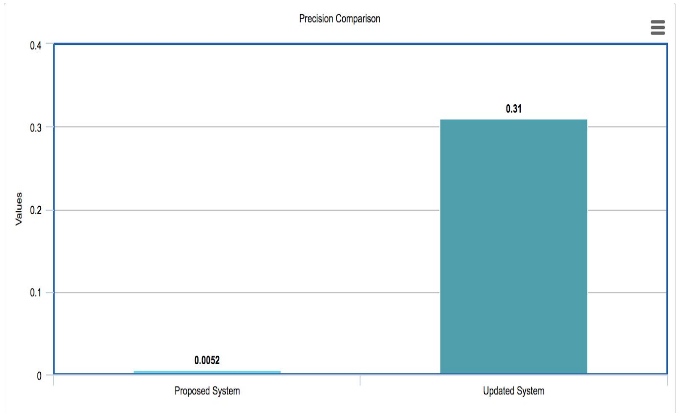
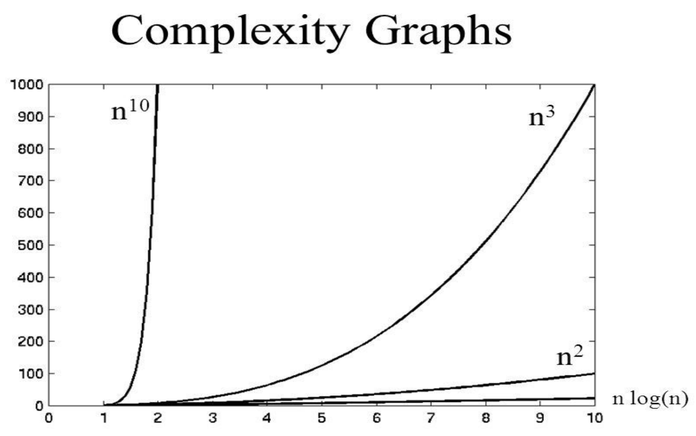


Fig 1

Computationally inferring the complexity of proposed system at worst is O(n3). While the updated system has a complexity of O(n2).

Fig 2[8]



Future Scope

Although the updated system is an improvement on the proposed system it still has significantly low precision rate. This is because word blend identification is a knowledge technology problem. It is just not the input of data to acquire a standard output. The values that are obtained as outputs are subject to the knowledge of what words are combined generally and have been utilised. Also there is a high probability of this dataset increasing in size as new and new terminoligies are coined everyday as the language is an evolving concept. That being understood it is still possible to improve the current system so as to identify the word blends that are used today frequently. One way to this is assignment of context. While above systems can blend the words, they can't distinguish between which words can possibly make the blend words together. This is because a context of their meaning is not provided, hence dine and inner can combine to form dinner according to above systems. However if a cluster of meanings is provided, then only words in associated clusters can combine to form blend words. Thus the difference in the meaning of the words dine and inner wouldn't allow it to be combined to form blend words. However even in above case words like bread and lunch can be combined to form brunch as opposed to the original breakfast and lunch combination. To mitigate this issue sentiment analysis can be used. The use of a given blend word in a sentence gives it the context of the original word. The sentence 'I am going to have my brunch at 12' is more likely to mean have breakfast and lunch combined at 12 than have bread and lunch at 12. Thus the word blend identification is an evolving procedure whose precision can be improved overtime but never can be equal to 100%.

References

[1] Grieve, J., Nini, A., & Guo, D. (2018). Mapping lexical innovation on American social media. *Journal of English Linguistics*, *46*(4), 293-319.

[2] 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

[3] 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

[4] Cook, P., & Stevenson, S. (2010). Automatically identifying the source words of lexical blends in English. *Computational Linguistics*, *36*(1), 129-149.

[5] P. Danziger, (2010). *Hamming Distance*. Retrieved from https://math.ryerson.ca/~danziger/professor/MTH108/Handouts/codes.pdf

[6] 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

[7] Kondrak, G. (2005, November). N-gram similarity and distance. In *International symposium on string processing and information retrieval* (pp. 115-126). Springer, Berlin, Heidelberg.

[8] Retrieved from https://www.google.com/search?q=complexity+n2+vs+n3&source=lnms&tbm=isch&sa=X&ved=0ahUKEwjA-5jAjs3kAhUYfX0KHTgcCHUQ_AUIEigB&biw=1440&bih=642#