Hashtag Segmentation: A Comparative Study Involving the Viterbi, Triangular Matrix and Word Breaker Algorithms

Samia F. Abd-hood[1,2] and Nazlia Omar[1]
[1] Faculty of Information Science and Technology, Universiti Kebangsaan Malaysia, Bangi, Malaysia
[2] Hadhramout University, Hadhramout, Yemen
Email: gp07233@siswa.ukm.edu.my, nazlia@ukm.edu.my

***Abstract*—Microblogging and social networking sites such as Twitter, Facebook, and Instagram, are becoming increasingly popular, registering more than 500 million posts each day. Twitter uses hashtags that are dynamic, user-generated text, preceded by a pound (#) symbol, to retrieve similar posts or topics, to mark events or to tag channels. Following segmentation, hashtags can be used for many Natural Language Processing (NLP) applications. These include sentiment analysis, text classification, named entity recognition, and sarcasm detection. This study delves into a comparison of three algorithms, namely the Viterbi, Triangular matrix and Word breaker algorithms, to determine the best among the three, for the segmentation of hashtags. These algorithms utilize different resources, to calculate the probability of the segmented parts, in order to rank the possible generated segmentations. For example, while the Viterbi and Triangular Matrix algorithms use two statistical corpora of unigram and bigram, the Word Breaker algorithm uses the n-gram language model. According to conducted experiment, the Viterbi algorithm is better for hashtag segmentation than the Triangular Matrix algorithm. This can be attributed to the manner in which the Viterbi algorithm conducts the backtracking. On the other hand, the Word Breaker algorithm, which can ascertain the meaningful tokens in the form of words, before proceeding with the segmentation of the remaining characters, is considered superior to both the Viterbi and Triangular Matrix algorithms, particularly when it comes to the detection of unknown words. Used together with the Good-Turing smoothing algorithm, the Word Breaker algorithm achieved 86.64% f1-score on a large language model.**

***Index Terms*—hashtag segmentation, twitter, viterbi algorithm, word breaker algorithm, triangular matrix algorithm**

I. Introduction

Social media platforms such as Twitter, Facebook and Instagram are currently inundated with information. The information provided by social media platforms, are used for a variety of applications, which include sentiment analysis and data mining [1], or for a user review for recommender systems [2]. The Twitter platform alone has 330 million monthly active users, who post half a billion Tweets each day [3]. Twitter data in the form of hashtags, play an essential role in several NLP applications. A hashtag is a user-generated text, used to label channels in the social media, for the delivery of a message to people, or for marking a significant event [4]. A hashtag is represented by the pound symbol (#), which is a form of metadata tag used on social networks, along with Twitter and a variety of micro-blogging services.

The hashtag concept is the brainchild of Chris Messina, who in a 2007 Tweet posted the message "How do you feel about using # (pound) for groups. As in #barcamp[msg]?". Initially looked upon as a "thing for nerds", the use of hashtags quickly became the norm among social media platforms.

Hashtags represented by one word (for instance *eclipse*) are called single-token hashtags, while hashtags represented by a combination of words without spaces are called multiple-token hashtags. Multiple-token hashtags can come in the lower case (for instance, *addictivetvshows*), with each token capitalized (for instance, *WakingUpTooEarly*), or with an underscore (for instance, *covid_19*). Hashtag segmentation involves the breaking down of a hashtag into its tokens. For example, following segmentation, the hashtag *addictivetvshows* becomes *addictive tv shows*.

Hashtag segmentation for NLP applications, has to do with the use of hashtag tokens, to automatically label sentiment analysis datasets with positive, negative, or neutral classifications, based on the sentiment of the hashtag [5]. Hashtag tokens are also used for identifying the names for named entity recognition systems [6], for classifying Tweets [7], and for detecting sarcasm [8].

This undertaking focuses on a comparative study involving the Viterbi, Triangular Matrix and Word Breaker algorithms, in terms of hashtag segmentation. This comparative study will serve to determine the most suitable algorithm, for the segmentation of Twitter data hashtags. The motivation for this study derives from the fact that (a) other hashtag segmentation techniques, such as the longest maximum matching, are ineffective for the detection of unknown words, (b) statistical approaches, such as the use of the language model to calculate probability, are more applicable for the identification of

Manuscript received January 19, 2021; revised August 25, 2021.

doi: 10.12720/jait.12.4.311-318

rarely used words, and (c) some hashtag segmentation approaches do not include an evaluation of the segmentation process itself. Belainine *et al.* used the segmented hashtag with other features, such as synonym and hypernymy from WordNet, for the classification of Tweets, not for the segmentation process [6]. The three algorithms selected for this study come with several plus points. The Viterbi algorithm has the capacity to monitor the segmented characters, generate new ones, and conduct matching with the previous as well as next characters, to determine if a correct segmentation has been generated. Additionally, the Viterbi algorithm can be employed with a unigram and bigrams in this study. The Word Breaker algorithm exploits the beam search algorithm to perform characterization by reducing the time and memory consumption to ensure that the optimal result is realized in a limited set. As reported by [9] and [10], the Word Breaker algorithm is considered robust for the segmentation process. As for the Triangular Matrix algorithm, while it has been previously employed for spell correction, there is no record of this algorithm being used for segmentation.

The rest of this paper is organized as follows: Section II describes previous works on hashtag segmentation, Section III explains the methodology of the segmentation process, Section IV presents the results and discussion, and Section V provides the conclusion to this paper.

II. Literature Review

The significance of the segmentation process derives from the lack of delimitation for languages such as Chinese, Japanese and Korean, as well as the frequent occurrence of abundant compounds in certain languages such as German. The segmentation concept has its roots in word segmentation for such languages. The methods used for word segmentation, often involve the tagging of the characters of the text, to identify the position of the character, in order to enter the boundaries of the words. Machine learning classifiers (as those used for the Chinese language segmentation) are then brought into the picture. While some researchers applied the neural network for tagging the characters, followed by the implementation of deep learning techniques to score the words [11], [12], others utilized the rule-based from corpora for the compound words (as for the German language). In a comparison exercise conducted by [13], involving a word-based model and a character-based model, the character-based model was proclaimed superior for hashtag segmentation.

The hashtag segmentation process is essential for many NLP applications. For instance, sentiment analysis relies on hashtags to label the dataset with positive, negative or neutral sentiments, as in [5].

Hashtag segmentation approaches can be grouped into four categories. Rule-based approaches identify the tokens of the hashtag, by detecting certain features included in the hashtag itself. For instance, the hashtag can be segmented through the detection of capitalization, numbers, or the underscore in the hashtag. However, as the dataset could also include hashtags in lowercase or uppercase patterns, the hashtags are segmented character by character, followed by the search for a match in a dictionary, to realize the correct segmentation. In a situation where there is more than one segmentation, the longest maximum matching technique is applied as in [6]. Likewise, [14] utilized the camel-cased technique to segment the hashtags of their dataset, and the POS tagging, to validate the English vocabulary included in a hashtag. While [8] applied the same process as [6] and [14] to segment the hashtags, unlike [6], they opted for the Viterbi-alike algorithm to select the best segmentation for lower case hashtags.

The word boundary detection approach considers hashtag segmentation a binary classification task. This approach entails the use of BI, which stands for Beginning or Inside the word schema. Each letter is considered an individual training instance for the learning algorithm, and labelled the first letter (boundary) of a word, or not, in the sequence. In order to determine the possible boundaries, some features are implemented to detect the (B)eginning or (I)nside character. The word boundary technique was demonstrated in studies conducted by [4] and [9]. They utilized vocabulary-based features, bigram-based features, and orthography-based features (for instance, the capitalized letters in a hashtag or a number), to determine the possible boundaries, then fed these instances into the Conditional Random Fields (CRFs) and Maximum Entropy (MaxEnt) classifiers, to realize segmentation.

Statistical approaches use the probability or other score functions, to rank the generated segmentations. The work in [15] involved the use of a regression model, to score the generated candidate segmentations produced by segmenting the hashtag, followed by matching either with Wikipedia pages, or the context of the Tweet. While [16] used a score function to rank the possible candidates, [7] used a probabilistic language model, to rank the candidates generated from the viterbi algorithm. The works of [17]-[19] all involved the use of the viterbi algorithm, with the former using it to segment the text of the Palm Leaf Manuscript, for Optical Character Recognition (OCR) systems. Ref. [18] implemented the Viterbi algorithm for named entity recognition systems, and [19] used it to segment the hashtags of Twitter for sentiment analysis systems. In contrast to this study, [18] implemented the viterbi algorithm with unigram corpus, to calculate probability.

Machine and deep learning approaches are frequently used for hashtag segmentation. In a study conducted by [20], a pairwise neural ranking technique was utilized to rank the segmentations generated by the word breaker algorithm. Researchers [21], [22] utilized the Recurrent Neural Network (RNN) to rank the segmentations. The work of [21] considered hashtag segmentation a sequence of labelling task, in which each symbol of a given string is labelled either 1 or 0. The value 1 represents a white space after this symbol, and 0, otherwise. Their approach was applied for the Russian language. Meanwhile, in a study conducted by [22], the Recurrent Neural Network (RNN) on character/byte was used, for the segmentation

of hashtags, with the utilization of the beam search algorithm.

III. METHODS

This section presents the framework of this study, and describes the methods utilized for hashtag segmentation. Also provided are explanations regarding the dataset and the pre-processing phase.

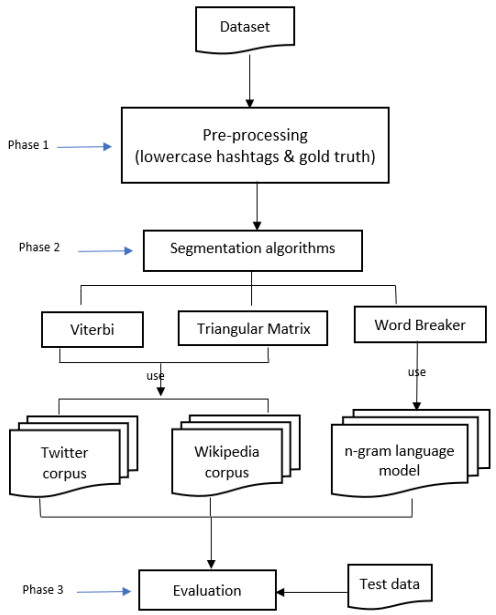

Figure 1. Hashtag segmentation framework.

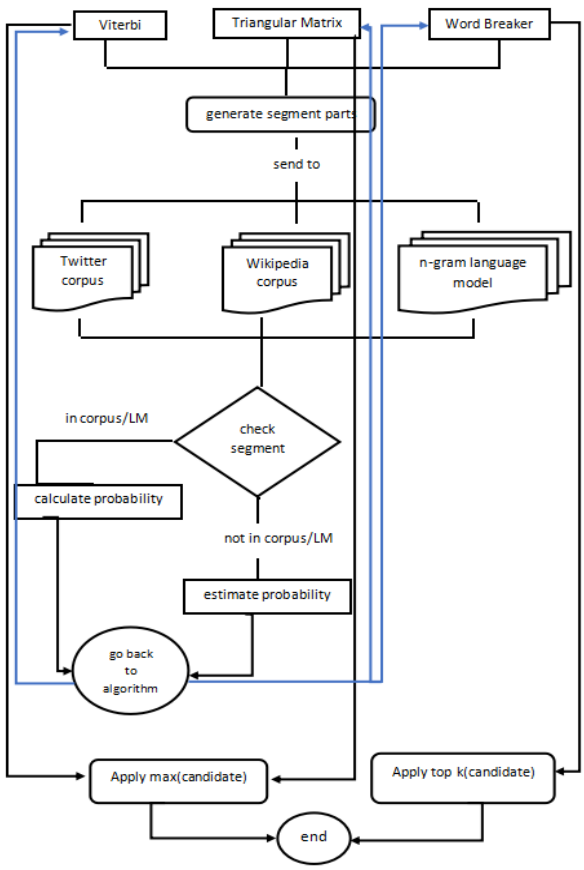

Figure 2. Probability calculation process.

Fig. 1 depicts the design of the automatic hashtag segmentation process, while Fig. 2 demonstrates the calculation process, for ascertaining the probability for the generated hashtag segments.

A. Dataset

The dataset is a fragment set introduced by [20], which originates from the Stanford dataset. It consists of 2518 hashtags, with their gold truth (segmentation performed manually). Only a single hashtag is extracted per Tweet. Eighty percent of dataset is for the train, and the rest is for the test. A sample of hashtags and their gold truth is displayed in Table I.

TABLE I. SAMPLE OF DATASET

Hashtag	Gold Truth
addictivetvshows	Addictive TV Shows
stupidaccident	stupid accident
FailWhaleBeGone	Fail Whale be gone, fail whale be gone, Fail Whale Be Gone
rssreader	RSS Reader

B. Pre-processing

This section describes the preparation of the dataset, for the segmentation phase. Only a single hashtag is extracted per Tweet. Also, the hashtag and gold truth are normalized to the lowercase. This serves to raise the evaluation quality, as prior to normalization, although the algorithms generated correct segmentations, due to case sensitivity, these segmentations do not match the gold truth. Table II illustrates the significance of gold truth normalization.

TABLE II. IMPORTANCE OF NORMALIZATION FOR GOLD TRUTH

Algorithm Segmentation	Gold Truth Segmentation	Segmentation Status
addictive tv shows	Addictive TV Shows	False
cloud views	Cloud Views	False

C. Resources Used for Segmentation

Several resources were used to conduct the experiments for this study. To begin with, two statistical corpora, one collected from Twitter, and the other from English Wikipedia, were utilized to calculate probability. Both corpora were collected by [19]. A unigram and bigram, with their frequencies, are used to calculate the probability of the Viterbi and Triangular Matrix segmented parts. During the segmentation process, the algorithms check for a match with the corpus. The detection of a match is followed by the return of the token as a segment. Otherwise, the probability is calculated as shown in Fig. 2. Two types of language models were used with the Word Breaker algorithm. One is a test.arpa[1] from KenLM [23], which is a small language model with size 2.71 KB, modified by way of the Kneser-Ney smoothing algorithm. This model consists of the unigram, bigram, trigram, 4-gram and 5-gram. The other type of

[1] https://github.com/zomux/modified-kenlm/tree/master/lm

language model used, which is larger, is taken from [20]. The language model was trained on 1.1 billion Tweets, with the Good-Turing smoothing algorithm, using SRILM [24]. The binary format of this model ensured that less space is taken up, and that the loading process is accelerated.

D. Segmentation Algorithms

The three hashtag segmentation algorithms investigated were the Viterbi, Triangular Matrix, and Word Breaker algorithms.

1) Viterbi algorithm: The Viterbi algorithm is a dynamic programming algorithm, designed to ascertain the most likely series of unknown finite states or hidden states (known as the viterbi path), that lead to a sequence of observed events. The objective of the Viterbi algorithm, is to uncover the best segmentation from possible generated ones, based on their probability. The Viterbi algorithm splits the hashtag into two parts, and then calculates the probability for the first part, using the statistical corpora collected from Twitter and Wikipedia separately. Subsequently, it applies a recursive function, to segment the remainder of the hashtag. Each time a new character enters the segmentation process, it is examined with the other characters in the segmentation process. **Algorithm 1** exhibits the pseudocode for the Viterbi algorithm.

Algorithm 1: Viterbi algorithm pseudocode

```
Input: hashtag
Output: segmented hashtag
if hashtag is not lowercase:
        apply regular expression to segment (camelCased_function ())
else:
        execute findSegment (hashtag, prev):
                if hashtag not text: return zero
                else:
                #segment the hashtag
                        for i in range(min(len(hashtag),
                maxSplitLength=20)):
                                first = hashtag[: i+1]
                                reminder = hashtag[i+1 : ]
                                return first, reminder
                        for first, reminder:
                                calculate the probability for first
                                recursive findSegment() for reminder
                                return candidates
                return max(candidates)
```

2) Triangular Matrix (TM): The Triangular Matrix algorithm, a particular type of square matrix, is widely used in the field of mathematics, notably in the linear algebra discipline. Better known for its spelling correction capacity in machine translation systems, this algorithm was put to the test for hashtag segmentation. TM breaks down the hashtag into small portions, to subsequently solve each piece once, before storing the result in an array. The next time when the algorithm recurs to segment the remainder of the hashtag, it determines if the portion was previously segmented. If so, instead of performing a re-computation, it proceeds to retrieve the best segmentation. This time-saving move renders the performance of the TM algorithm quicker.

The TM algorithm computes the different segmentations of a prefix of the substring, at a specific position of the inner loop. It starts to segment the input hashtag from the left, and when the inner loop reaches the end of the string, the last segment is retrieved. Subsequently, the segmentation process begins on the remainder of the input from the second character. As shown in Fig. 3, each time a new character starts the segmentation process, the first character is excluded. The pseudocode for the algorithm is illustrated in **Algorithm 2**.

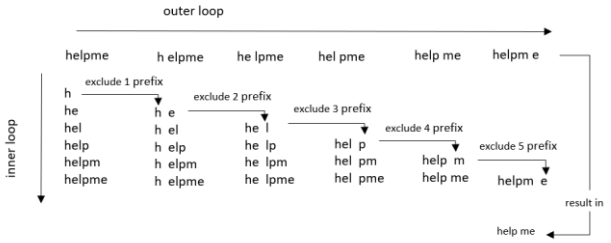

Figure 3. Triangular Matrix segmentation process.

Algorithm 2: Triangular Matrix pseudocode

```
        input: hashtag
        output: segmented hashtag
        initialize arraySize = 0
        initialize arrayWidth = 0
        segmentedSpaceBits = [] #to store spaces instead of
segmented text
        findSegment(hashtag):
                #i represents all possible part start position (outer loop)
                for i in range(0, len(hashtag)):
                        if i > 0:
                                combine best segmentation of i with
                        part_1 generated from the inner loop
                                imax = min(len(hashtag)-i,
                        maxSegWordLen)
                                for j in range(1, len(imax)):
                                        part_1.split(i, j)
                                        calculate probability then the best
                                segmentation stores in array
                                        #end inner loop
                        return the best segmentation for outer loop based on its
        probability
```

3) Word Breaker (WB): The implementation of the WB algorithm entails the use of the beam search algorithm, with the n-gram language model. The beam search is a heuristic algorithm that explores all the possible candidates, and returns the results based on some heuristic information. The beam search algorithm considers the segmented hashtag a tree. It analyses the input hashtag one character (token) at a time from the left, and determines if a word boundary token in that position is appropriate. If so, two segmented nodes of the hashtag will be generated. As the algorithm processes the entire data, the best k number of candidates will be selected from the current possibly made. K in this experiment is set up to 10, meaning that the algorithm will generate 10 different segmentations. When a new character enters the segmentation process, the algorithm examines it with other characters in the segmentation process, to identify the possible boundary for the segmentation. **Algorithm 3** illustrates the pseudocode for the algorithm.

Algorithm 3: Word Breaker pseudocode

```
input: hashtag
output: segmented hashtag
initialize positionCurrentWord = 0
initialize score = 0
initialize segments = null
beamSearch(hashtag):
    for i in range(positionCurrentWord +1 to len(hashtag)+1):
        currentWord = hashtag[positionCurrentWord :i]
        getScore(segments, currentWord):

            calculate probability using language model
        push the new segment in list
        while list not empty:
            if newSegment.position == len(hashtag):
                push newSegment in topk_segments
            else:
                recursive beamSearch(newSegment)
for segment in topk_segments:
    return segment[k]
```

E. Evaluation

The performance of the segmentation process was evaluated by way of the four standard evaluation metrics: accuracy, precision, recall and f-score. Accuracy represents full matching with the gold truth. It is a percentage of corrected segmentation with respect to whole instances in the dataset [20].

$$\text{accuracy} = \frac{\text{number of correctly segmented hashtag}}{\text{whole number of instances of dataset}} \quad (1)$$

Precision, recall, and f-score consider the partial matches on the segmentation level [20]. Precision represents the percentage of correct words in the segmentation, to the number of words in the segmentation by an algorithm.

$$\text{Precision} = \frac{\text{number of correct words in a segmentation}}{\text{length of the segmentation}} \quad (2)$$

Recall represents the number of correct words in segmentation to the gold truth.

$$\text{Recall} = \frac{\text{number of correct words in a segmentation}}{\text{length of the gold truth segmentation}} \quad (3)$$

F-score is the harmonic measure between precision and recall.

$$\text{F-score} = \frac{2 * \text{precision} * \text{recall}}{\text{precision} + \text{recall}} \quad (4)$$

IV. Results and Discussion

The experiments were performed using 2518 hashtags with their gold truth segmentation (manual segmentation). The dataset comprised hashtags of English Twitter data. The experiments were conducted to assess the performance of three algorithms (Viterbi, Triangular Matrix and Word Breaker algorithms) in order to determine the best hashtag segmentation algorithm. Python 3.7 was used for experiments on Anaconda software. The Viterbi and Triangular Matrix algorithms use the unigram and bigram for probability calculations. The Word Breaker algorithm utilizes two forms of the n-gram language model to estimate likelihood; one smoothed with Good-Turing, and the other smoothed with the Kneser-Ney technique.

A. Evaluation Results for Each Algorithm

This section discusses the performance of each algorithm with regards to hashtag segmentation. The first comparison for the Twitter and Wikipedia corpora was performed for the Viterbi algorithm, while the second was performed for the Triangular Matrix, as it uses the same corpora as the Viterbi algorithm. The Word Breaker algorithm was compared to two types of language models: one smoothed with the Good-Turing algorithm, and the other smoothed with Kneser-Ney. As the WB algorithm generates k different segmentations, it was examined in different positions. For this experiment, the k value set up to 10, in order to obtain 10 different segmentations. The details of these experiments are provided below.

1) Viterbi algorithm: The results from the experiment on the Viterbi algorithm, using the Wikipedia and Twitter corpora, are exhibited in Table III. As can be observed, the English-Wikipedia corpus outperformed the Twitter corpus by small portions, in all evaluation metrics for the test set, as the former includes more tokens in the hashtag after splitting. This denotes a correct segmentation for the algorithm on this corpus.

TABLE III. Results in the Percentage of Viterbi Algorithm

The test set with normalization				
Statistical corpora	**Accuracy**	**Precision**	**Recall**	**F1-score**
Twitter	76.83	81.17	78.01	78.99
English-Wikipedia	80.31	82.94	79.41	80.54

2) Triangular Matrix algorithm: The details of the experiment conducted on the Twitter and Wikipedia corpora, for the Triangular Matrix algorithm, are displayed in Table IV. Here again, the English-Wikipedia corpus outperformed the Twitter corpus, for the same reasons stated for the Viterbi algorithm.

TABLE IV. Results in the Percentage of Triangular Matrix Algorithm

The test set with normalization				
Statistical corpora	**Accuracy**	**Precision**	**Recall**	**F1-score**
Twitter	71.04	74.11	69.07	70.55
English-Wikipedia	72.97	75.14	70.07	71.53

3) Word Breaker algorithm: The Word Breaker algorithm was examined for two types of Language Models (LMs). One type is an n-gram language model, which is a small sized language model smoothed with Kneser-Ney (KN). The latter is 3-gram, large in scale, and smoothed with the Good-Turing (GT) algorithm. The WB algorithm was tested on different positions of the segmentation, as this algorithm returns the top k segmentations, where k is set up to 10, unlike the Viterbi and TM algorithms, which return only the best one according to its probability. Table V displays the results

for the WB algorithm, on the 1st position of the test set, for both language models.

TABLE V. RESULTS IN PERCENTAGE FOR THE WORD BREAKER ALGORITHM

The test set on the 1st position of segmentation for two language models				
Language Model	**Accuracy**	**Precision**	**Recall**	**F1-score**
Small (KN)	55.79	55.79	55.79	55.79
Large (GT)	75.67	78.11	79.09	78.48

KN: Kneser-Ney
GT: Good-Turing

Table VI shows the results for the WB algorithm in the 5th position of segmentation.

TABLE VI. RESULTS IN PERCENTAGE FOR THE WB ALGORITHM IN THE 5TH POSITION OF SEGMENTATION

The test set on 5th position of segmentation for two language models				
Language Model	**Accuracy**	**Precision**	**Recall**	**F1-score**
Small (KN)	69.30	74.13	72.20	72.89
Large (GT)	81.08	83.08	83.95	83.43

Table VII shows the results for the WB algorithm in the 10th position of segmentation.

TABLE VII. RESULTS IN PERCENTAGE FOR THE WB ALGORITHM IN THE 10TH POSITION OF SEGMENTATION

The test set on the 10th position of segmentation for two language model				
Language Model	**Accuracy**	**Precision**	**Recall**	**F1-score**
Small (KN)	83.01	89.86	87.16	88.07
Large (GT)	84.36	86.30	87.13	86.64

As shown in Table V and Table VI, the large model with GT outperformed the small one, which is smoothed with KN in the first and fifth positions. This is an indication that in terms of segmentation, the larger model is superior. Also, it is obvious, that increasing the position, covers more generated segmentations. This enhances the effectiveness of the algorithm, for the detection of the best segmentation. The need to cover several generated segmentations has to do with overcome the nature of the beam search algorithm, which can either detect the optimal target immediately, or reach the end of the search with nothing to show. Surprisingly, the small model, with KN, outperformed the larger one, with GT, for precision and f-score. This can be attributed to the superior capacity of KN for the detection of abbreviations/rare words, and single words in the test dataset. GT, on the other hand, performs better when it comes to multi-token hashtags.

B. Comparison among the Three Algorithms

The first comparison exercise involved the Viterbi and Triangular Matrix algorithms. Both use the same statistical corpora.

Fig. 4 displays the comparison between the Viterbi and TM algorithms, for the English-Wikipedia corpus. Fig. 5 shows a similar comparison, conducted on a statistical corpus collected from Twitter.

According to Fig. 4 and Fig. 5, the Viterbi algorithm outperformed the TM algorithm. This can be attributed to the nature of the former, which examines a new character with other characters each time it enters the segmentation process, and computes the probability every time. This form of backtracking, provides the Viterbi algorithm with an edge over the TM algorithm.

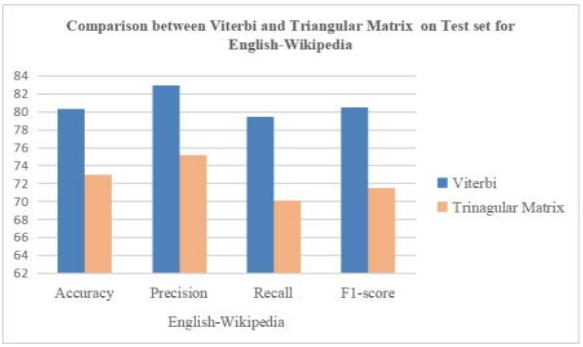

Figure 4. Viterbi vs. Triangular Matrix for the English-Wikipedia corpus.

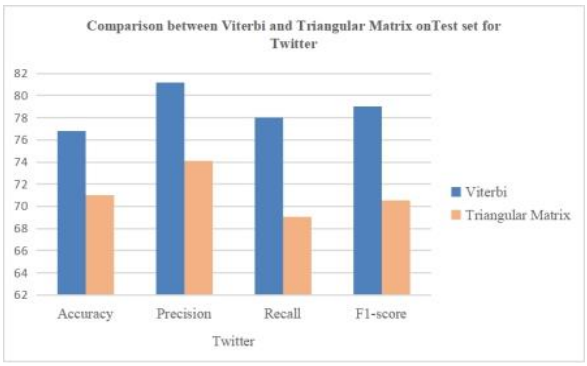

Figure 5. Viterbi vs. Triangular Matrix for Twitter corpus.

The second comparison exercise, which was conducted on the Word Breaker algorithm, involved the use of different types of language models. Fig. 6 demonstrates the manner in which the language model, smoothed with Good-Turing, outperformed the language model, modified with the use of the Kneser-Ney smoothing algorithm.

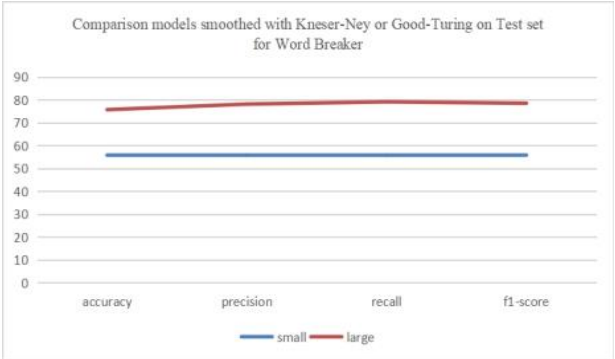

Figure 6. Large LM (GT) vs. small LM (KN) for WB.

As the Word Breaker algorithm selects the top 10 segmentations, from the possible generated segmentations, an additional comparison was applied to the language model smoothed with Good-Turing, in different positions of the segmentation. Fig. 7 shows the results derived from this third comparison exercise, with regards to these different positions.

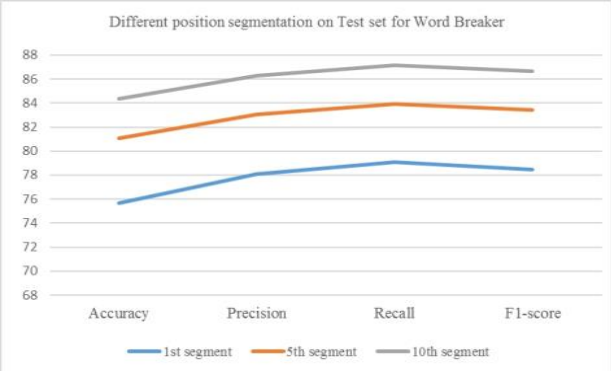

Figure 7. Segmentations of WB for large LM (GT) on different positions.

The fourth comparison exercise involved the best results attained by the Viterbi, Triangular Matrix and Word Breaker algorithms, on various resources. As can be observed in Fig. 8, the WB algorithm, with an f1-score of 86.64%, outperformed both the Viterbi algorithm (f1-score of 80.54%) and the Triangular Matrix algorithm (f1-score of 71.53%).

This outcome can be attributed to the capacity of the WB algorithm, to detect the meaningful tokens in the segmentation process, as it uses heuristic information, and can calculate their probabilities regarding the other tokens in the segmentation. Moreover, due to the smoothing technique employed, the language model can identify unknown words better, even if only one word from the segmentations is apparent.

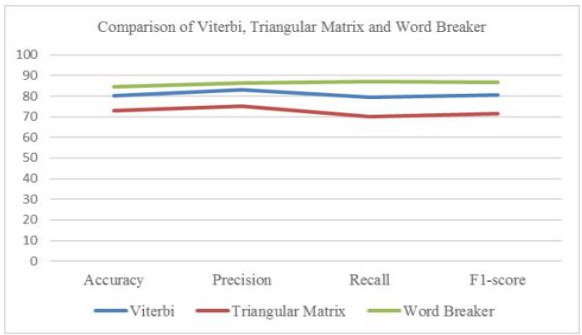

Figure 8. Viterbi vs. WB vs. TM algorithms.

V. Conclusion

The main goal of this study is to select the best algorithm among the Viterbi, Triangular Matrix, and Word Breaker algorithms, for hashtag segmentation on the Twitter dataset. To begin with, the Viterbi algorithm outperformed the Triangular Matrix algorithm, for both the Twitter and Wikipedia word statistical corpora used.

As this investigation involved the use of different types of language models, a separate comparison exercise was conducted, to determine the best model for the WB algorithm, when it comes to the segmentation of Twitter hashtags. The findings revealed that the most outstanding performance was delivered by the language model, smoothed with the Good-Turing algorithm, as it proved to be best at detecting multi-token hashtags.

To summarize, the Word Breaker algorithm, which is implemented using the beam search algorithm with the n-gram language model, proved to be superior to the other two algorithms, investigated during this undertaking. In terms of f1-score, the Word Breaker algorithm achieved 86.64%, compared to 80.54% for the Viterbi algorithm, and 71.53% for the Triangular Matrix algorithm.

In view of the results derived through this comparative study, the Word Breaker algorithm can be considered superior to the Viterbi and Triangular Matrix algorithms, when it comes to hashtag segmentation. For future work, different datasets should be examined for the algorithms, so that the results can be generalized. A large language model should be considered for the Word Breaker algorithm, as this can serve to enhance its segmentation performance. And lastly, machine learning techniques should be brought into the picture, to improve the hashtag segmentation process.

Conflict of Interest

The authors declare no conflict of interest.

Author Contributions

For the research work, Samia F. Abd-hood carried out the investigation on techniques of hashtag segmentation, implemented the algorithms and performed the experiment. Nazlia Omar advised the investigation and writing of this manuscript, in which all authors had approved the final version.

Acknowledgment

This project is funded by UKM under the research code GP-2020-K007009. The first author would like to thank Establishment of Hadhramout for Development and Humanity.

References

[1] B. Saberi and S. Saad, "Sentiment analysis or opinion mining: A review," *Int. J. Adv. Sci. Eng. Inf. Technol.*, vol. 7, no. 5, pp. 1660-1666, 2017.

[2] S. M. Al-Ghuribi and S. A. M. Noah, "Multi-criteria review-based recommender system-The state of the art," *IEEE Access*, vol. 7, pp. 169446-169468, 2019.

[3] A. Omnicore. (2020). Twitter-statistics. [Online]. Available: https://www.omnicoreagency.com/twitter-statistics/

[4] A. Çelebi and A. Özgür, "Segmenting hashtags and analyzing their grammatical structure," *J. Assoc. Inf. Sci. Technol.*, vol. 69, no. 5, pp. 675-686, 2018.

[5] I. Alfina, D. Sigmawaty, F. Nurhidayati, and A. N. Hidayanto, "Utilizing hashtags for sentiment analysis of tweets in the political domain," in *Proc. 9th Int. Conf. Mach. Learn. Comput.*, 2017, pp. 43-47.

[6] B. Belainine, A. Fonseca, and F. Sadat, "Named entity recognition and hashtag decomposition to improve the classification of tweets," in *Proc. 2nd Work. Noisy User-generated Text*, 2016, pp. 102-111.

[7] C. Simeon, H. J. Hamilton, and R. J. Hilderman, "Word segmentation algorithms with lexical resources for hashtag classification," in *Proc. IEEE Int. Conf. Data Sci. Adv. Anal., IEEE*, 2016, pp. 743-751.
[8] D. G. Maynard and M. A. Greenwood, "Who cares about sarcastic tweets ? Investigating the impact of sarcasm on sentiment analysis," in *Lr. 2014 Proceedings*, ELRA, 2014, pp. 4238-4243.
[9] A. Celebi and A. Özgür, "Segmenting hashtags using automatically created training data," in *Proc. Tenth Int. Conf. Lang. Resour. Eval.*, 2016, pp. 2981-2985.
[10] P. Bansal, R. Bansal, and V. Varma, "Towards deep semantic analysis of hashtags," in *Proc. Eur. Conf. Inf. Retr.*, Springer, Cham., 2015, pp. 453-464, 2015.
[11] D. Cai and H. Zhao, "Neural word segmentation learning for chinese," arXiv Prepr. arXiv1606.04300, 2016.
[12] J. Ma, K. Ganchev, and D. Weiss, "State-of-the-Art Chinese word segmentation with Bi-LSTMs," arXiv Prepr. arXiv1808.06511, 2018.
[13] X. Li, Y. Meng, X. Sun, Q. Han, A. Yuan, and J. Li, "Is word segmentation necessary for deep learning of Chinese," arXiv Prepr. arXiv1905.05526, 2019.
[14] T. Declerck and P. Lendvai, "Processing and normalizing hashtags," in *Proc. Int. Conf. Recent Adv. Nat. Lang. Process.*, 2015, pp. 104-109.
[15] P. Bansal, S. Jain, and V. Varma, "Towards semantic retrieval of hashtags in microblogs," in *Proc. 24th Int. Conf. World Wide Web*, 2015, pp. 7-8.
[16] J. Reuter, J. Pereira-Martins, and J. Kalita, "Segmenting Twitter hashtags," *Intl. J. Nat. Lang. Comput.*, vol. 5, no. 4, pp. 23-36, 2016.
[17] G. Peng, P. Yu, H. Li, and L. He, "Text line segmentation using Viterbi algorithm for the palm leaf manuscripts of Dai," in Proc. *Int. Conf. Audio, Lang. Image Process, IEEE*, 2016, pp. 336-340.
[18] S. P. Sharmila and P. K. Sujatha, "Segmentation based representation for tweet hashtag," in *Proc. Seventh Int. Conf. Adv. Comput., IEEE*, 2016, pp. 1-7.
[19] C. Baziotis, N. Pelekis, and C. Doulkeridis, "Datastories at semeval-2017 task 4: Deep lstm with attention for message-level and topic-based sentiment analysis," in *Proc. 11th Int. Work. Semant. Eval.*, 2017, pp. 747-754.
[20] M. Maddela, W. Xu, and D. Preoţiuc-Pietro, "Multi-task pairwise neural ranking for hashtag segmentation," arXiv Prepr. arXiv1906.00790, pp. 2538-2549, 2019.
[21] T. Glushkova and E. Artemova, "Char-RNN and active learning for hashtag segmentation," arXiv Prepr. arXiv1911.03270, pp. 1-15, 2019.
[22] Y. Doval and C. Gómez-Rodríguez, "Comparing neural- and n-gram-based language models for word segmentation," *J. Assoc. Inf. Sci. Technol.*, vol. 70, no. 2, pp. 187-197, 2019.
[23] K. Heafield, "KenLM : Faster and smaller language model queries," in *Proc. Sixth Work. Stat. Mach. Transl.*, 2011, no. 2009, pp. 187-197.
[24] A. Stolcke, "SRILM - An extensible language modeling toolkit," in *Proc. 7th Int. Conf. Spok. Lang. Process*, 2002, pp. 901-904.

Samia F. Abd-hood received her BSc degree in Computer Science from Hadhramout University, Hadhramout, Yemen in 2011. She has completed her MSc in Information Science from Universiti Kebangsaan Malaysia (UKM). Her major field in natural language processing and machine learning. She is an assistant lecturer at Hadhramout University, Yemen.

Nazlia Omar is currently an Associate Professor at the Center for AI Technology (CAIT), Faculty of Information Science and Technology (FTSM), Universiti Kebangsaan Malaysia (UKM). She holds her PhD from the University of Ulster, UK. Her main research interest is in the area of Natural Language Processing and Computational Linguistics. She is a member of the Asian Language Processing Lab (ASLAN) at CAIT, FTSM, UKM.