Segmenting Hashtags and Analyzing Their Grammatical Structure

Arda Çelebi and Arzucan Özgür

Department of Computer Engineering
Boğaziçi University
Bebek, 34342 İstanbul, Turkey
{ arda.celebi, arzucan.ozgur } @boun.edu.tr

This is a preprint of an article published in "Çelebi, A., Özgür, A. 'Segmenting Hashtags and Analyzing Their Grammatical Structure.' Journal of the Association for Information Science and Technology (JASIST). DOI: 10.1002/asi.23989" http://onlinelibrary.wiley.com/journal/10.1002/(ISSN)2330-1643

Abstract

Originated as a label to mark specific tweets, hashtags are increasingly used to convey messages that people like to see in the trending hashtags list. Complex noun phrases and even sentences can be turned into a hashtag. Breaking hashtags into their words is a challenging task due to the irregular and compact nature of the language used in Twitter. In this study, we investigate feature-based machine learning and language model (LM) based approaches for hashtag segmentation. Our results show that LM alone is not successful at segmenting non-trivial hashtags. However, when the N-best LM-based segmentations are incorporated as features into the feature-based approach, along with context-based features proposed in this study, state-of-the-art results in hashtag segmentation are achieved. In addition, we provide an analysis of over two million distinct hashtags, auto-segmented by using our best configuration. The analysis reveals that half of all 60 million hashtag occurrences contain multiple words and 80% of sentiment is trapped inside multi-word hashtags, justifying the need for hashtag segmentation. Furthermore, we analyze the grammatical structure of hashtags by parsing them and observe that 77% of the hashtags are noun-based, whereas 11.9% are verb-based.

1. Introduction

After one year Twitter was debut in 2006, on August 23, 2007 at 10:25pm, Chris Messina suggested something that no one ever has suggested before. He tweeted "how do you feel about using # (pound) for groups. As in #barcamp [msg]?" This was the birth of hashtags. Their existence came out of necessity to tag channels in Twitter so that people can filter those tweets that are labeled by specific hashtags. Since then, we use hashtags not only to label channels, but also to convey the actual message that we want people to hear. When big events happen, Twitter users get coordinated and use hashtags that are specific to that event. That way, frequently mentioned hashtags in a short time period show up in the trending hashtags list, thus the messages that these hashtags convey. Some hashtags are short and compact, while others, especially the ones that convey some message, can be long, particularly when they are formed from clauses or complete sentences. *#IfTheyGunnedMeDown*, *#TwitterIsBlockedInTurkey*, *#BringBackOurGirls* are example hashtags that were seen in the trending lists in the past years. As these examples suggest, in order to do complete social network content analysis, we should be able to segment hashtags into their original words and unleash their content for Natural Language Processing (NLP) tasks such as sentiment analysis and named-entity recognition.

Considering the history of the NLP field, hashtags are quite a new concept. Recently, a number of studies have shown that they can be effectively used for various social media NLP tasks such as for text classification (Billal et al., 2016), query expansion (Bansal et al., 2015b), and emotion detection (Qadir and Riloff, 2014). These studies reveal that hashtags have started to attract the attention of the NLP community. However, so far, no extensive study has been done on segmenting hashtags and analyzing their grammatical structure. Most prior studies use their function of being a label without breaking them into their constituent words (Stilo and Velardi, 2017). When they need segmented hashtags, they either use the traditional word segmentation tools (Chen et al., 2016) or employ simple glossary and rule based approaches (Billal et al., 2016). Despite their high accuracy at segmenting words, such tools have not been originally designed for hashtag segmentation. Bansal et al. (2015a) and Celebi and Ozgur (2016) showed that hashtag segmentation presents a more challenging task than traditional word segmentation, as both studies outperformed the state-of-the-art word segmentation tool Word Breaker (Wang et al., 2011) at segmenting hashtags.

In our previous study, we developed a feature-based approach that makes use of vocabulary and orthography-based features to segment hashtags (Celebi and Ozgur, 2016). In this study, we went further with our previous research by adding the following three aspects:

1. Most prior work do the segmentation in isolation from the context in which a hashtag occurs. In this study, we introduce context features that utilize the words occurring together with the hashtag.

2. Apart from the feature-based approach, we present a language model (LM) based hashtag segmentation approach. We look into how LM alone performs at segmenting hashtags and then, blend the LM-based approach into the feature-based one by introducing new features based on the top best LM-based segmentations. We show that combining context-based features and LM-based features leads to improved performance.

3. We run our best hashtag segmentor on 2.1 million distinct hashtags and obtain their automatically segmented forms. With that set, we first measure how much sentiment is trapped inside multi-word hashtags. Then, we parse this set by using a dependency parser and study their grammatical structure. To the best of our knowledge, this is the first study that reports on the constituent word analysis of a large set of automatically segmented hashtags. We argue that such an analysis reveals the extend of hashtags' complexity as well as the need to use hashtag segmentors.

2. Related Work

Precursor to hashtag segmentation, the earliest related work is word segmentation. Thanks to languages like Chinese and Arabic, where there are no spaces between words, word segmentation is one of the most studied fields in the literature. The simplest approach for word segmentation is using a dictionary to find the longest matching word in a given input by applying the maximum matching algorithm or its variations (Wong and Chan, 1996). Many state-of-the-art systems employ statistical approaches, as they are better at handling unknown words and ambiguous cases. Probabilistic methods based on Hidden Markov Models (HMMs), Maximum Entropy (MaxEnt) models, and Conditional Random Fields (CRFs) have been used for word segmentation (Peng et al., 2004; Xue, 2003). Discriminative models such as word-based perceptron algorithm (Zhang and Clark, 2008), neural networks (Rumelhart and McClelland, 1986), lazy learning approaches (Daelemans et al., 1997), and unsupervised methods (Chen et al., 2012) have also been used in this domain.

Another approach for word segmentation is using a language model to find the best possible word segmentation of a given sequence of characters among many possible segmentations. Lee et al. (2003) use trigram LM to determine the best possible morpheme sequence for Arabic word segmentation. In a web-scale word segmentation study, Wang et al. (2011) show that they can unify and generalize word breaking techniques under the Bayesian minimum risk framework which can consider multiple document styles with minimal heuristics. Their tool, namely Word Breaker achieves an accuracy of 97.18% with a trigram model.

Compared to traditional word segmentation studies, there are only a handful of studies on hashtag segmentation. Srinivasan et al. (2012) use an unsupervised method with a joint probability model learned from multiple corpora. However, they evaluate their segmentor only by observing improvement in the Twitter search recall measure. Berardi et al. (2011) tackle the problem as a compound word segmentation task and apply the well-known Viterbi algorithm (Jr., 1973) to choose the best possible word sequence. Their hashtag segmentor has not been evaluated independently, but has been used as a component in a search system for tweets. Maynard and Greenwood (2014) develop a hashtag tokenizer for GATE (Cunningham et al., 2002) by applying a Viterbi-like algorithm to look for the best possible match by using multiple gazetteers in the GATE framework. Declerck and Lendvai (2015) focus on normalizing the surface forms of hashtags in the form of case normalization, lemmatization and syntactic segmentation. They examine tagged and parsed versions of *CamelCased* hashtags in a domain specific data set. Bansal et al. (2015a) introduce a hashtag segmentor which starts with a set of possible segmentations and ranks them by using five features, including context similarity. They achieve 87.3% accuracy using 5-fold cross validation on their manually annotated test set. In this study, we also use their test set, which we call Test-STAN. In our recent study, we developed a feature-rich approach using MaxEnt and CRFs as the learning algorithms (Celebi and Ozgur, 2016). Various vocabulary- and orthographic-based features were designed. The use of auto-segmented hashtags for training versus tweets was evaluated. While using tweets for training resulted in better performance, MaxEnt was shown to outperform CRFs. An accuracy of 88.2% was achieved on our manually annotated test set, Test-BOUN[1]. On Test-STAN, the best accuracy value obtained was 85.4%.

3. SEGMENTING HASHTAGS

In the following sections, we present our feature-based and LM based approaches to segment hashtags, as well as their combination through LM based features. Then, we describe our data sets and experimental results.

[1]Our project website is at http://tabilab.cmpe.boun.edu.tr/projects/hashtag_segmentation/

3.1. Methods

Feature-based Approach

We formulate the segmentation problem as a boundary detection task in a given sequence of characters. In this setup, each character becomes an individual training instance for the learning algorithm and is labeled as being the first character (boundary) of a word or not in the sequence. Imagine we have an imaginary cursor pointing to a character in the given sequence as in Figure 1. At each position, we determine the active features and the learning algorithm uses these features to model this binary classification task. For each training instance, its class and the list of active features are given to a MaxEnt model for learning[2]. Each feature is represented as a string constructed as feature_name=feature_value, where feature value is obtained based on the current cursor position. We consider various features with different characteristics. These features and their initialized values based on the cursor position shown in Figure 1 are listed in Table 3. The first column shows how to set the values for the corresponding features by using functions[3]. The second column contains the active feature instances, each one constructed as a string consisting of feature name and value pairs. We group the features into four categories:

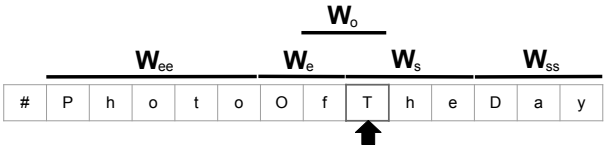

Figure 1: Showing the words detected around the cursor position for the hashtag #PhotoOfTheDay; W_s: longest word at cursor position; W_e: longest word before W_s; W_{ee}: longest word before W_e; W_{ss}: longest word after W_s; W_o: overpassing word "ft"

Vocabulary-based features look for words around the cursor position in a character sequence based on a given vocabulary. We create our vocabulary from all used training data sets, which are described in the Training and Test Data Sets section. As shown in Figure 1, we look at five positions w.r.t. the cursor position for the longest matching words[4]. While the existence of an overpassing word (W_o) such as "ft" in Figure 1 suppresses the boundary decision, words starting at the cursor position (W_s) and ending just before the cursor position (W_e) are positive indicators for a boundary for that position. The longer they are, the more likely there is a boundary. Apart from using the longest word W_s as a feature, we also define features that represent words in terms of their lengths (*len*($\cdot$)) and floored negative unigram log probabilities[5] (*flNegLogProb*($\cdot$)). For example, the unigram log probability of the word "the" is -1.808. Hence, one of the features in Table 3 represents the word "the" (W_s) with the combination of its length 3 and its floored negative log probability 1 (i.e., $length\_and\_floored\_negative\_log\_probability\_of\_longest\_word\_at\_cursor = 3 + 1$). We also use the class codes of the words (*wordClass(*$\cdot$*)*), which we obtain from CMU's Twitter Word Clusters (Owoputi et al., 2012). To give an example, since the words "of" and "the" are represented with Word Cluster codes 10110 and 110100, respectively, one of the features in Table 3 (i.e., $word\_class\_bigram\_around\_cursor = 10110 + 110100$) uses this information to represent the word bigram, where the first word (W_e) occurs before the cursor position and the next one (W_s) starts at the cursor position. Such representations group words into equivalence classes and reduces the number of model parameters, hence reducing the complexity of the model. We observed that especially features where words are represented with their lengths and floored negative log probabilities together are more effective in general compared to using the words themselves. Moreover, since short words[6] like "of", "the" etc. may be seen very frequently inside any hashtag, one of our effective features looks for a short word *shortWordAt(*$\cdot$*)* at cursor position which is also supported by the existence of surrounding words W_e and W_{ss}.

Ngram-based features are extensions of word-based features. Detecting word bigrams around the cursor position like (W_e, W_s) and (W_s, W_{ss}) can be quite indicative of a boundary. Moreover, we define features based on the bigram frequencies obtained from all our training data sets. These frequencies are clustered according to their ranges and represented by their cluster indices. *bigramFreqClass(*$\cdot$*,* $\cdot$*)* assigns a cluster index of 1, for example, if frequency is higher than 500[7]. Similarly

[2]Available at http://homepages.inf.ed.ac.uk/lzhang10/maxent_toolkit.html.

[3]The descriptions of the functions that are not already described in text are as follows: *ifExist*($\cdot$) checks if the given input exists; *isUpper*($\cdot$) and *isLower*($\cdot$) check if there is an upper and lower case letter in the given position, respectively; *isNumber*($\cdot$) checks if there is a digit in the given position; *ifRestIsNumber*($\cdot$) checks if all characters from the given position till the end of the input are digits; *countRepeatedCharactersAt*($\cdot$) counts the number of repeating character starting at the given position and *countRepeatedCharactersAtBackward*($\cdot$) does the same backward; *ifShortWord*($\cdot$) checks if a given word is a short word; *ifBigramExists*($\cdot$, $\cdot$) checks if the given two words are seen as a bigram in the training set.

[4]Assume that there is no longer word in our vocabulary that can match in this specific example.

[5]We use Microsoft's Web N-Gram Service to collect unigram log probabilities of words in our vocabulary.

[6]The list of short words and short bigrams (consisting of short words) that we compiled is available at our project web site.

[7]Cluster indices of 6, 5, 4, 3, 2, and 1 are assigned when frequency is lower than 5, 10, 50, 100, 200, and 500, respectively.

TEMPLATE OF FEATURE VALUE	ACTIVE FEATURE INSTANCE (feature_name=feature_value)
W_s	longest_word_starting_at_cursor=the
len(W_o)	length_of_overpassing_word=2
ifExists(W_o)	if_overpassing_word_exists=TRUE
ifExists(W_e&W_s)	if_words_around_cursor_position_exist=TRUE
ifExists(W_s)&*isUpper*(*cursor*[0])&*isLower*(*cursor*[-1])	if_word_at_cursor_position_exists_and_cursor_position_in_upper_case=TRUE
ifExists(W_e&W_s)&*isUpper*(*cursor*[0])&*isLower*(*cursor*[-1])	if_words_around_cursor_position_exist_and_cursor_position_in_upper_case=TRUE
len(W_s)	length_of_longest_word_starting_at_cursor=3
len(W_o)+*flNegLogProb*(W_o)	length_and_floored_neg_log_probability_of_overpassing_word=2+3
len(W_s)+*flNegLogProb*(W_s)	length_and_floored_neg_log_probability_of_longest_word_at_cursor=3+1
len(W_e)+*len*(W_s)+*len*(W_o)	length_of_words_around_cursor_and_overpassing_word=2+3+2
shortWordAt(*cursor*[0])+*len*(W_e)+*len*(W_{ss})	short_word_in_middle_and_length_of_surr_words=the+2+3
len(W_e)+*flNegLogProb*(W_e)	length_and_floored_neg_log_probability_of_word_ended_just_before_cursor=2+1
len(W_{ee})+*flNegLogProb*(W_{ee})+*len*(W_e)+*flNegLogProb*(W_e)	length_and_floored_neg_log_probability_of_two_words_before_cursor=5+3+2+1
len(W_e)+*len*(W_s)	length_of_longest_words_around_cursor=2+3
len(W_e)+*flNegLogProb*(W_e)+*len*(W_s)+*flNegLogProb*(W_s)	length_and_floored_neg_log_probability_of_words_around_cursor=2+1+3+1
len(W_s)+*flNegLogProb*(W_s)+*len*(W_{ss})+*flNegLogProb*(W_{ss})	length_and_floored_neg_log_probability_of_bigram__starting_at_cursor=3+1+3+1
flNegLogProb(W_e)+*flNegLogProb*(W_s)	floored_neg_log_probability_of_words_around_cursor=1+1
wordClass(W_e)+*wordClass*(W_s)	word_class_bigram_around_cursor=10110+110100
wordClass(W_s)+*wordClass*(W_{ss})	word_class_bigram_starting_at_cursor=110100+11100111010
shortNgramAround(*cursor*[0])+*len*(W_{ee})+*len*(W_{ss})	short_ngram_over_cursor_and_length_of_surrounding_words=of_the+5+3
bigramFreqClass(W_e,W_s)+*flNegLogProb*(W_o)+*len*(W_o)	freq_class_of_bigram_around_cursor_and_overpassing_word_length=1+3+2
ifBigramExists(W_e,W_s)	if_words_around_cursor_seen_as_bigram=TRUE
bigramFreqClass(W_e,W_s)	frequency_class_of_bigram_around_cursor=1
bigramFreqClass(W_s,W_{ss})	frequency_class_of_bigram_starting_at_cursor=1
bigramFreqClass(W_s,W_{ss})+*bigramFreqClass*(W_e,W_s)	frequency_classes_of_bigram_starting_at_cursor_and_around_cursor=1+1
len(W_e)+*len*(W_s)+*len*(W_o)+*bigramFreqClass*(W_e,W_s)	len_of_words_around_cursor_and_overpassing_and_freq_class_of_bigram_around_cursor=2+3+2+1
W_e+W_s	bigram_with_short_word_around_cursor=of+the
cursor[0]+*cursor*[1]+*cursor*[2]	three_character_starting_at_cursor=t+h+e
orth(*cursor*[-1])+*orth*(*cursor*[0])	orthographic_shape_of_previous_and_current_character=c+C
orth(*cursor*[-1])+*orth*(*cursor*[0])+*orth*(*cursor*[1])	orthographic_shape_of_previous_current_and_next_characters=c+C+c
orth(*cursor*[-2])+*orth*(*cursor*[-1])+*orth*(*cursor*[0])	orthographic_shape_of_previous_two_and_current_characters=C+c+C
orth(*cursor*[-3])+*orth*(*cursor*[-2])+*orth*(*cursor*[-1])+*orth*(*cursor*[0])	orthographic_shape_of_previous_three_and_current_characters=c+C+c+C
isNumber(*cursor*[0])&*ifRestIsNumber*(*cursor*[0])	if_number_starts_at_cursor_position_and_continues_till_end=FALSE
countRepeatedCharactersAt(*cursor*[0])	num_of_times_character_at_cursor_repeats_forward=0
countRepeatedCharactersAtBackward(*cursor*[0])	num_of_times_character_at_cursor_repeats_backward=0

Table 1: Listing all active vocabulary-, ngram- and orthography-based feature instances generated based on the cursor position shown in Figure 1, where W_s=the, W_{ss}=day, W_e=of, W_{ee}=photo, and W_o=ft. The plus sign (+) is used as a separator to make the reading easy.

to the *shortWordAt(·)* feature, there is also a feature that looks for bigrams that are constructed from short words only, like "of the". *shortNgramAround(·)* returns such bigram around the cursor position.

Orthography-based features complement the vocabulary-based ones. *orth*(·) converts characters to their orthographic shapes[8]. We use three of them as listed in Table 3. A capitalized letter or a number around the cursor position can be a good indicator of a boundary. Orthography-based features are especially effective when no word is detected at the cursor position.

Context-based features make use of words that occur in the same tweet that hosts the hashtag. We consider three context-based features. Table 2 presents the context feature instances generated for the example tweets in Figure 2. The simplest feature looks at whether a word in the tweet is seen in the hashtag starting at the cursor position. To give an example, in Figure 2, the first tweet includes a hashtag #nationalwineday, while the word "wine" also occurs in the tweet. Since such detection doesn't require any vocabulary, this feature is complementary to vocabulary-based features.

The second feature looks for semantically similar words in the tweet to the word starting at the cursor position (W_s) in the hashtag. If W_s is semantically similar to a word in the tweet, then that signals the existence of W_s in the hashtag. We

[8]Upper-case letter is replaced with 'C', lower-case one with 'c', digit with 'd'; otherwise the character itself is used

TEMPLATE OF FEATURE VALUE	ACTIVE FEATURE INSTANCE (feature_name=feature_value)
len(W_s)+*flNegLogProb*(W_s)	word-in-context=4+3
orthoShapeDist(W_s)	ortho-shape-dist-in-context=cCc-cc
cosine-similarity(W_s, W_c)	highest-sem-similarity-in-context=0.7

Table 2: Listing all context-based feature instances generated based on the cursor positions shown in Figure 2, where W_s=wine for Tweet 1, W_s=blocked for Tweet 2. The plus sign (+) is used as a separator to make the reading easy.

use the distributed representations of words to compute their semantic similarity. To obtain that, we train the *word2vec* tool (Mikolov et al., 2013) on the Stanford Twitter Sentiment Analysis Dataset[9] with default parameters to produce a 100-dimensional vector for each word. The value of this feature is set as the cosine similarity value between the vectors of W_s and the word that is most similar to W_s in the tweet. In Figure 2, the word "prevent" in Tweet 2 can help to identify the boundary starting with the semantically similar word "blocked".

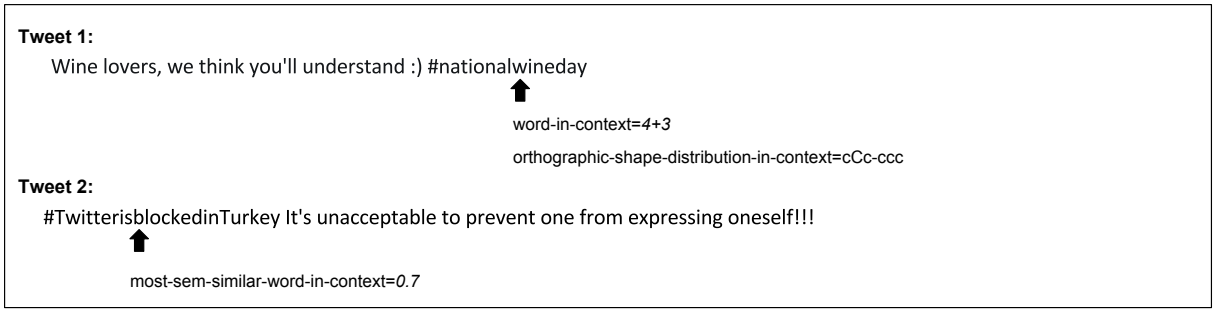

Figure 2: Sample tweets exemplifying the context-based features.

Considering the fact that the same hashtag may occur in a number of tweets, we can collect a set of tweets that contain the same hashtag and use all their words as an extended context. We call this global context, whereas the case where only one tweet that hosts the hashtag is used is called local context. This increases the chance of identifying words in hashtags that are mentioned in the global context or semantically related to them. Moreover, since different capitalized versions of the same hashtag may occur in a given tweet set, as a third feature, we look for orthographic shape differences among all occurrences of a hashtag. We count the number of different capitalization cases around the cursor position in a window of 3 characters, order them based on their frequencies, and create a feature from these ordered orthographic shapes. For example, assume that we have 10 tweets that include #NationalWineDay and 2 tweets with #nationalwineday. As the cursor is at the starting character of word 'wine', there are 10 'cCc' and 2 'ccc' orthographic shapes observed around that cursor position. For this case, the feature has the value 'cCc-ccc', which is a concatenation of the ordered orthographic shapes with a hyphen in between.

LM-based Approach

Besides the feature-based approach to detect word boundaries, we also consider a completely different approach, which takes into account all likely segmentations, scores each with a language model (LM), and chooses the highest scoring word sequence as the best segmentation for the input character sequence. An LM-based approach is generally successful at detecting word sequences and such a capability is particularly helpful when it comes to detecting short words (e.g. "in", "at" etc.) between other words. This is where the feature-based approach struggles. Nevertheless, the performance of the LM-based approach is hindered by unknown words as they disrupt the true word sequence, in which case, LM inevitably scores an incorrect word sequence as the best possible segmentation. Considering that tweets are full of typos and frequently introduced new words, the LM-based approach may not perform as expected in this domain. One way to cope with this is to use as much data as possible to train the LM. Hence, we generate the training dataset using randomly selected tweets from the SNAP data set[10]. The SNAP data set contains 20-30% of all public tweets posted during the seven months period between June and December, 2009. It includes 476 million tweets. While creating the training set, we filter out the hashtags, links, mentions and punctuations, which leaves out only word sequences. In order to investigate the effect of training set size on results, we created multiple training sets with increasing sizes, starting from 10 million tokens up to 1 billion tokens. From each training set, we generate the language model with the SRILM tool. We generate the LM from 4-grams and prune the vocabulary to contain the most frequent 1M words. We use Kneser-Ney smoothing as the

[9]Visit http://help.sentiment140.com/for-students/ for more information.
[10]Visit https://snap.stanford.edu/data/twitter7.html for more information.

discounting option and use the default values for the rest of the parameters. In our experiments, we use the learned LM models with the OpenFST tool[11], which considers all likely segmentations for a given input character sequence and selects the highest scoring segmentation as the best based on the used model.

LM-based Features

Considering the noisy nature of tweets, the LM-based approach might not always find the exact segmentation. However, if we consider the N-best segmentations, instead of the best suggested one, it is more likely to have the correct (or very close to correct) segmentation among them. We argue that the N-best segmentations can be a supplementary data for the feature-based approach and we consider three features that make use of that data as boundary clues. This effectively combines the feature-based and LM-based approaches. First, we use the rank of the highest scored segmentation that has a word boundary at the cursor position. Secondly, as an extension to the first one, we concatenate the ranks of all segmentations that have a boundary at the cursor position. Thirdly, we use the number of segmentations that have a boundary at the cursor position as the feature value. To illustrate these with an example, Figure 3 shows the top 10 highest scoring segmentations of "greatmovie", highest at the top. Alongside, the active feature instance for each LM-based feature is listed based on the shown cursor position in this example. The equal sign separates the name of the feature and its value.

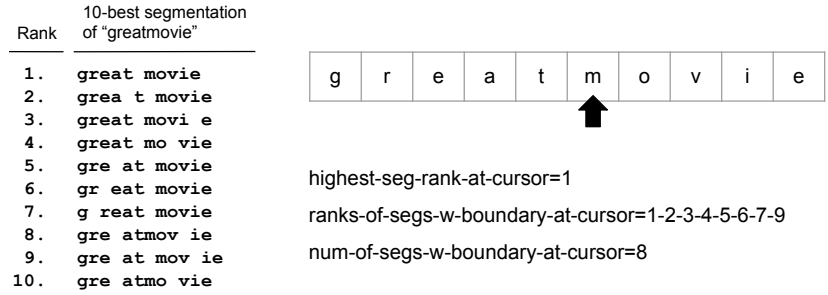

Figure 3: Showing the values of LM-based features when the cursor is at the 6th position while segmenting "greatmovie."

TEMPLATE OF FEATURE VALUE	ACTIVE FEATURE INSTANCE (feature_name=feature_value)
highestSegmentRankAt(*N-bests*, i)	highest-seg-rank-at-cursor=1
segmentRanksAt(*N-Bests*, i)	ranks-of-segs-w-boundary-at-cursor=1-2-3-4-5-6-7-9
numOfSegmentsAt(*N-Bests*, i)	num-of-segs-w-boundary-at-cursor=8

Table 3: Listing all LM-based feature instances generated based on the cursor position shown in Figure 3, where i is the position of the cursor and *N-bests* holds the N-best LM-produced segmentations of the input.

In case of the first feature, the rank represents the index of the segmentation in the N-best list and the higher it is (i.e., the smaller the index), the more likely that there is a boundary at that position. For the other two features, the more segmentations have the boundary at the cursor, again, the more likely that there is an actual boundary there. We observe the state-of-the-art results when all three LM features are used together.

3.2. Training and Test Data Sets

In order to learn how to segment words, learning algorithms need training data which include already segmented words. To make our results comparable, we use the data sets used in (Celebi and Ozgur, 2016) and (Bansal et al., 2015a). After filtering the links and other non-literal tokens in tweets, boundaries of the words were used as gold standard for training. Two tweet data sets were generated. The first one (Tw-STAN) includes tweets from the Stanford Twitter Sentiment Analysis Dataset, which were collected by querying for emoticons only. The second one (Tw-BOUN) was collected via Twitter Search API by specifically querying for names of well-known actors, sports people, politicians, movies, tv shows, sports teams, and companies. Thus, this data set can be considered to be specific to certain domains. The difference between the collection methods for the two data sets allows us to measure the effect of "almost" randomly collected tweets (Tw-STAN) compared to relatively more domain-centric tweets (Tw-BOUN) on performance.

Our third training dataset, which is a hashtag data set (HASHTAGS), was obtained from the SNAP Twitter data set by employing a rule-based segmentation method. After collecting hashtags from SNAP data set, we searched for consecutive

[11]We use *make-ngram-pfsg* and *pfsg-to-fsm* tools to make the output of the SRILM tool compatible with OpenFST. Visit http://www.openfst.org for more information.

word sequences in the SNAP tweets such that their concatenation corresponds to one of those hashtags in case insensitive manner and without considering whether the hashtag occurs in the same tweet. For 1.25M hashtags, we detected at least one word sequence. To given an example, for the #twittermarketing hashtag, we detected 29,892 occurrences of the word bigram "twitter marketing" and 116 occurrences as a single word "twittermarketing." We selected the most frequent word sequence for each hashtag, only if the hashtag's total occurrence is higher than 10 and the selected word sequence corresponds to 75% of the total occurrences of its all seen word sequences. This heuristic generated 803K automatically segmented hashtags, which is called the HASHTAGS data set.

In our trainings, we use portions of these three datasets with increasing sizes. Table 4 lists the number of tokens in those datasets, which roughly gives the number of positive training instances for hashtag segmentation. Note also that the column at the right hand-side of the table indicates the number of hashtags selected from the HASHTAGS dataset to make the token counts compatible with the tweet data sets.

# of Tweets	# of Tokens			# of Hashtags
	Tw-BOUN	Tw-STAN	HASHTAGS	
5K	56K	60K	57K	24K
10K	115K	118K	118K	49K
20K	237K	249K	237K	98K
50K	594K	602K	595K	246K
100K	1189K	1210K	1180K	489K

Table 4: Number of tokens in Tw-BOUN, Tw-STAN, and HASHTAGS datasets with increasing sizes.

For testing purposes, we use the two manually segmented hashtag sets used in (Celebi and Ozgur, 2016). The first set includes 2268 randomly selected hashtags from the Stanford Twitter Sentiment Analysis Dataset. 1268 of these were manually segmented by Bansal et al. (2015a) and called as Test-STAN, whereas the rest, which is called Dev-STAN, was randomly selected from the same data set and manually segmented by Celebi and Ozgur (2016). The second set consists of Dev-BOUN and Test-BOUN data sets, each having 500 manually segmented hashtags which were selected randomly from Tw-BOUN (Celebi and Ozgur, 2016). In this study, we use Dev-STAN and Dev-BOUN as development sets, and Test-STAN and Test-BOUN as test sets. Similar to the difference between Tw-STAN and Tw-BOUN, these two test sets have different characteristics. STAN test sets are more likely to contain complex hashtags with high randomness, whereas BOUN test sets contain more domain-centric hashtags with less randomness. It would be interesting to observe how the segmentor performs in both cases.

3.3. Experiments

Baselines

For both test sets, we consider three approaches at different strengths as baselines. The F-score and accuracy results are given in Table 5. Accuracy shows what percentage of hashtags are segmented completely correctly. On the other hand, F-score is the harmonic mean of precision and recall, where precision shows what percentage of outputted words is correct with respect to the test set and recall is what percentage of actual words in the test set is identified correctly. Our first baseline is based on HMM which was trained on character tri-grams. It considers both the current and previous two characters with respect to the cursor position for the boundary detection. We trained it with 100K randomly selected tweets from Tw-BOUN. However, it performed poorly. As a second approach, we use another off-the-shelf tool, called Hashtag Tokenizer (Maynard and Greenwood, 2014) in the GATE framework (Cunningham et al., 2002). This gazeteer-based approach achieves slightly better performance than the previous baseline results, yet still obtains low performance.

Approach	Test-BOUN		Test-STAN	
	F_1-score	Acc.	F_1-score	Acc.
HMM	74.0	69.3	63.0	64.3
GATE Hashtag Tokenizer	76.5	70.2	73.3	71.5
Word Breaker	**84.4**	**86.2**	**84.6**	**83.6**

Table 5: Baseline results on Test-BOUN and Test-STAN sets.

While the previous two approaches can be considered as weak baselines, the third approach, Microsoft's Word Breaker (Wang et al., 2011), provides a strong baseline, considering that it outperforms the other baselines significantly in all

metrics. Hence, we consider it as our actual baseline. As shown by Bansal et al. (2015b), hashtags can present a more challenging segmentation task for Word Breaker, since it has originally been designed to break URLs.

Context-based Results

Context-based features require us to collect tweets for both training and test hashtags. However, we observe that some hashtags in the test sets occur rarely. In order to measure the ideal effect of context and its size correctly, we need to make sure that all hashtags in the test sets have the same number of tweets in their global context. To do that, from our training sets, we collected all tweets that contain hashtags from our test sets. The number of hashtags in Test-BOUN and Test-STAN that occurred in at least 100 tweets was 300 and 500, respectively. We call these new sets Test-BOUN-300 and Test-STAN-500 and use them to evaluate the effects of context features.

When we train the best (BEST) feature combination from (Celebi and Ozgur, 2016) on the 100K HASHTAGS dataset without adding any context-based features, the accuracies on Test-BOUN-300 and Test-STAN-500 are calculated as 90.0% and 88.9%, respectively. As we add each context feature into this best feature combination, we measure how much increase in accuracy each context feature provides as shown in Table 6 and Table 7. We start the size of the context from one tweet (C=1), which we consider as the local context case. Then, we gradually increase the number of tweets up to 100 tweets (C=100).

Features	C=1	C=5	C=10	C=20	C=50	C=100
BEST + Word in Context (WIC)	90.0	90.0	90.3	90.0	90.3	90.7
BEST + Sem. Similar Word in Context (SSWIC)	89.7	90.3	89.7	90.3	90.3	90.3
BEST + Orthographic Shape Distribution in Context (OSDIC)	-	91.3	90.3	90.7	90.7	90.0
BEST + WIC + SSWIC + OSDIC	89.7	91.3	91.3	91.7^a	91.3	**92.7**^{a}

Table 6: Accuracy on the Test-BOUN-300 test set, while the baseline (BEST) accuracy where no context is used is 90.0%.

In case of Test-BOUN-300, we achieve the highest scores when we use all context-based features. Even though using 100 tweets as global context results in the highest accuracy of 92.7%, using as few as 20 tweets still outperforms the baseline accuracy, which is 90.0%, statistically significantly[12]. On the other hand, with Test-STAN-500, even though all context features in general result in improvement, the orthographic shape distribution in context (OSDIC) feature results in the best performance. These results suggest that even if we can collect as few as 20 tweets for global context purposes, it can still increase the segmentation accuracy.

Features	C=1	C=5	C=10	C=20	C=50	C=100
BEST + Word in Context (WIC)	90.1	89.5	90.4	90.1	89.7	89.9
BEST + Sem. Similar Word in Context (SSWIC)	90.6	90.4	89.7	90.1	90.1	90.1
BEST + Orthographic Shape Distribution in Context (OSDIC)	-	90.8	**91.6**^{a}	**91.6**^{a}	91.4	91.0
BEST + WIC + SSWIC + OSDIC	89.9	91.4^b	90.8	90.6	90.8	90.6

Table 7: Accuracy on the Test-STAN-500 test set, while the baseline (BEST) accuracy where no context is used is 89.9%.

LM-based Results

The LM-based results for different training set sizes are shown in Table 8. LM achieves an accuracy of 90% on Test-BOUN when trained with 1B tokens. This score is higher than the best previously reported feature-based result of 88.2% (Celebi and Ozgur, 2016). On the other hand, the best LM-based accuracy obtained on Test-STAN (80.4%) is lower than the best previously reported feature-based accuracy (85.4%), it is even lower than the accuracy of the Word Breaker baseline (83.6%). One way to explain the difference between the two test sets is to look at the perplexity scores[13] of those test sets. Perplexity measures how successful LM is at predicting the next word in a sequence. The lower the score is, the better the LM is at predicting the next word. For Test-BOUN and Test-STAN, the perplexity scores are 750 and 6920, respectively. The huge difference in these scores explains why LM fails on Test-STAN.

Even if the most likely segmentation returned by LM is not the correct one, it is likely that the correct segmentation is among the top N segmentations produced by LM. In Tables 9 and 10, we calculate the accuracy of LM at top N, by

[12]In Tables 6 and 7, "a" and "b" indicate statistically significant result compared to the baseline for $p<0.05$ and $p<0.1$ (based on a paired two-tail t-Test), respectively.

[13]We use 100M-token LM for perplexity calculation. We report ppl1 value given by SRILM's *ngram* tool as the perplexity score.

LM Training Size	Test-BOUN		Test-STAN	
in # of tokens	F_1-score	Acc.	F_1-score	Acc.
10M	89.3	84.6	81.0	78.0
100M	92.2	88.4	82.4	79.7
1B	**93.2**	**90.0**	**82.9**	**80.4**

Table 8: Results of the best (top 1) LM-based word segmentation on Test-BOUN and Test-STAN sets.

considering the result as correct, if the gold standard segmentation is among the top scored N segmentations. F-score is computed by considering the segmentation with the lowest edit distance to the gold standard. The accuracy increases up to 94.8% and 93.2% on Test-BOUN and Test-STAN, respectively. On both data sets, the best segmentation is most of the time in the top 2. These results indicate that the top N segmentations contain valuable clues for segmentation.

TopN	**LM Training Data Size in # of tokens**					
	10M	100M	1B	10M	100M	1B
	F_1-Score at Top N			Accuracy at Top N		
N=1	89.3	92.2	93.2	84.6	88.4	90.0
N=2	93.1	95.8	96.2	89.8	93.6	94.4
N=5	93.4	96.2	96.6	90.0	94.0	94.8
N=10	93.4	96.2	96.6	90.0	94.0	94.8

Table 9: Best possible results in Top N on Test-BOUN.

TopN	**LM Training Data Size in # of tokens**					
	10M	100M	1B	10M	100M	1B
	F_1-Score at Top N			Accuracy at Top N		
N=1	81.0	82.4	82.9	78.0	79.7	80.4
N=2	87.2	91.4	92.9	84.8	89.8	91.6
N=5	87.7	92.7	94.4	85.3	91.2	93.2
N=10	87.8	92.9	94.4	85.4	91.5	93.2

Table 10: Best possible results in Top N on Test-STAN.

Results with LM-based and Context Features Combined

As described in the LM-based Features section, we designed three LM-based features using the 10-best LM segmentations. Due to run-time constraints, we used LM trained on 100M tokens, rather than 1B. Table 11 shows how much increase these LM-based features add onto the best feature combination obtained on the Test-BOUN set in (Celebi and Ozgur, 2016). For any type of training set, we can observe up to 5.6 points increase in accuracy and 3.9 points increase in F_1 score on Test-BOUN test set. Especially the increase in case of the HASHTAGS training set is consistently high. Note that there is no statistically significant difference between using all context features or using only the OSDIC context-based feature.

Training Set	**LM-based features used?**	**Training Data Size**									
		5K	10K	20K	50K	100K	5K	10K	20K	50K	100K
		F_1-score					Accuracy				
Tw-BOUN	No	90.5	91.3	91.4	91.5	91.5	85.6	86.8	87.2	87.4	87.4
	Yes	93.5	94.6	93.9	93.6	94.1	90.2	91.8	90.4	90.0	90.8
Tw-STAN	No	92.0	92.1	92.1	92.4	91.8	88.0	88.2	87.8	88.2	87.6
	Yes	93.8	93.3	93.7	93.6	93.1	90.4	89.6	90.2	90.2	89.2
HASHTAGS	No	89.9	89.4	89.7	90.8	91.0	85.8	85.0	85.0	86.2	86.6
	Yes	93.4	92.3	93.6	94.4	94.6	90.2	88.6	90.6	91.4	91.8
	Yes + OSDIC	93.2	88.3	93.1	**94.9**	93.9	89.8	83.0	90.0	**92.2**	90.6
	Yes + All-C	93.5	93.6	94.3	94.5	93.9	90.4	90.4	91.6	91.6	90.8

Table 11: Best results on Test-BOUN set. Baseline (MS Word Breaker) F_1-score = 84.4%, Accuracy = 86.2%

Likewise, Table 12 depicts a similar situation for the Test-STAN set. Using the LM-based features with 10-best segmentations improves the accuracy by up to 3 points. The increase is relatively lower than what we observe in Test-BOUN. This is again due to the fact that LM is better at detecting word sequences in TEST-BOUN than in Test-STAN. Similarly, using LM-based features on the HASHTAGS training set does not bring that much improvement compared to the 5.6-points increase on the Test-BOUN set. Again, in case of the HASHTAGS training set, as we add context-based features, we observe up to 3 points improvements in accuracy. The best accuracy in Table 12 is 88.5%, which is higher than the best previously reported value of 85.4% (Celebi and Ozgur, 2016). To make this score comparable with the accuracy of 87.3% obtained by Bansal et al. (2015a), we recalculated it on their original dataset which includes over 100 duplicate hashtags compared to our Test-STAN. In that case, our best score increases to 88.8%[14].

[14]The difference between 88.8 and 87.3 is statistically significant with p<0.05 based on one sample t-Test.

Training Set	LM-based features used?	Training Data Size									
		5K	10K	20K	50K	100K	5K	10K	20K	50K	100K
		F_1-score					Accuracy				
Tw-BOUN	No	84.9	86.9	87.0	87.0	87.0	83.0	85.1	85.4	85.3	85.4
	Yes	88.2	89.3	90.0	89.9	**90.2**	86.1	87.6	88.4	88.3	**88.5**
Tw-STAN	No	84.4	86.4	85.8	85.9	86.7	83.9	84.2	83.5	84.8	84.8
	Yes	86.8	87.8	88.1	88.3	87.9	84.8	85.8	86.3	86.5	85.8
HASHTAGS	No	84.8	85.3	85.4	85.8	86.2	82.9	83.5	83.7	84.1	84.5
	Yes	85.4	84.6	86.9	87.6	86.8	83.5	82.4	85.2	85.8	84.9
	Yes + OSDIC	86.7	87.3	88.1	87.8	89.5	84.8	85.4	86.5	86.1	87.9
	Yes + All-C	86.6	85.8	87.1	87.5	89.2	84.7	83.8	85.4	85.7	87.6

Table 12: Best results on Test-STAN set. Baseline (MS Word Breaker) F_1-score = 84.6%, Accuracy = 83.6%

Additionally, in both Table 11 and Table 12, the segmentors trained with auto-segmented hashtags (HASHTAGS) tend to produce better results as the size of the training set increases. Such behavior is less consistent with the other types of training sets.

Error Analysis

In order to better understand in which cases our hashtag segmentor fails, we examine the output of the highest scoring configuration[15] from Table 12 on the Dev-STAN set. We observe that almost all of mis-segmented hashtags contain only a single misclassified boundary. 42% of the erroneous cases are due to special words with low frequency, such as foreign words or proper names. 23% involve incorrectly segmented ordinary words like *outdoorsport*. The next most seen error type is caused by numeric expressions in single word names such as *o2007comp* with 13% of coverage. Similarly, 8% of errors are caused by capitalized characters inside single word names, such as *AbleGamers*. Such cases are very difficult to handle unless that word is seen in the training data. Otherwise, the segmentor tends to break the words from the capitalized characters. More surprisingly, 9.5% of errors are caused by lack of apostrophe in hashtags. For example, if a hashtag contains the string "thats" corresponding to "that's", the manually segmented form in the gold standard is "that s". In such cases, the segmentor fails to separate the contracted form of a word from the word that it is attached to.

4. Structural Analysis of Auto-segmented Hashtags

In this section, we investigate the importance of having a hashtag segmentor. We first count the number of words in millions of auto-segmented hashtags and then measure how much of the sentiment is trapped inside multi-word hashtags. Then, we analyze the grammatical structure of auto-segmented hashtags to observe the complexity of hashtags. In these experiments, we use the hashtags extracted from the SNAP tweet data set and apply our best model to get their segmented versions. We calculate these statistics on both 2.1 million distinct hashtags[16] and all 60 million hashtag occurrences in the SNAP data set.

4.1. Word Count in Auto-segmented Hashtags

When we look at the word count calculated on the set of distinct hashtags in Table 13, 87.7% of the hashtags consist of multiple words, which means that hashtags are not simple one-word labels. Even when we calculate the percentages in all 60 million occurrences, this value only drops to 48.6%. In other words, half of the time we need a hashtag segmentor to break down a hashtag into its constituent words. Most of distinct multi-word hashtags contain two or three words and people tend to use hashtags that are no longer than three words in 93.8% of all cases.

4.2. Trapped Sentiment inside Multi-word Hashtags

Considering that half of the hashtag occurrences consist of multiple words, it is important to measure how often a word with sentiment occurs in multi-word hashtags. We use the AFINN sentiment analysis tool[17] (Nielsen, 2011), which assigns

[15]The one that achieves 88.5% accuracy when it is trained on 100K tweets from Tw-BOUN.
[16]The count is taken in case-insensitive mode.
[17]Available at https://github.com/fnielsen/afinn

# of Words	% in Distinct 2.1M Hashtags	% in All 60M Occurrences
1	12.3	51.4
2	38.5	30.8
3	24.7	11.6
4	12.5	4.3
5	6.1	1.3
> 5	5.9	0.6

Table 13: Number of words in auto-segmented hashtags.

a sentiment score to a given text.

Sentiment	% in Distinct 2.1M Hashtags	% in All 60M Occurrences
Neutral	75.7	86.8
Positive	10.7	7.1
Negative	13.6	6.1

Table 14: Percentage of observed sentiment in auto-segmented hashtags.

Table 14 shows that, out of 2.1M distinct hashtags, 10.7% convey positive sentiment and 13.6% have negative sentiment. When we do the same calculation on all hashtag occurrences, the percentages drop to 7.1% and 6.1%, respectively. One thing to point in Table 14 is that people tend to use positive hashtags more often than negative hashtags, yet there are more distinct negative hashtags than positive ones. In other words, people are more creative at creating negative hashtags.

Our further analysis of hashtags containing positive or negative sentiment reveals that only 0.5% of distinct hashtags with positive sentiment, and 0.8% of distinct hashtags with negative sentiment are single-word, and the remaining ones are multi-word. When we consider all hashtag occurrences, we observe that only 20.7% of positive hashtags and 19.5% of negative hashtags are single-word. This means that, only around 20% of sentiment (either positive or negative) are seen in single-word hashtags. To put it another way, around 80% of sentiment is trapped inside multi-word hashtags.

4.3. Parsing Auto-segmented Hashtags

As the language in tweets are noisy and irregular, we chose to use TweeboParser (Kong et al., 2014) to parse segmented hashtags. TweeboParser is a dependency parser which is originally trained to parse tweets. It outputs which word in the given input grammatically depends on which other word, along with the part-of-speech (POS) tag assigned to each word. A word that does not depend on any other word is called root. Figure 4 shows an example dependency parse tree for the "Definition of Fail" hashtag. It is a noun phrase which includes the noun "Definition" as the root (or head) of the phrase and has an attached prepositional phrase which is headed by the preposition "of" and its dependent noun "Fail".

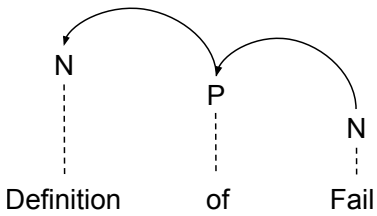

Figure 4: Dependency parse tree for "Definition of Fail" where the arrows point which word depends on which other.

While the dependency parser outputs the grammatical structure of the entire sentence or phrase, to keep our analysis simple, we only consider the root of the whole parse and its dependent tags one level below. When we just look at the root tag, it tells us which type of phrase or clause the whole structure is. When we look at its dependent tags, we basically observe how it is made of. While most of the hashtags have single root like the one shown in Figure 4, there can also be multiple roots, where each root corresponds to an independent clause. However, for the sake of simplicity we perform our analysis on hashtags with single root in their parses.

Analysis of Root Tags

Table 15 shows the percentages of the most frequent POS tags at the root level from the parses of segmented hashtags. One might argue that most hashtags are supposed to be noun phrases, since they are mostly used for labeling. However, it indicates that almost half of the 2.1M distinct hashtags are noun headed expressions, namely noun phrases. Following that, with 26.0%, verb headed expressions come in the form of various clauses. As we consider all occurrences, we observe that noun-based expressions are used more often (77.1%) and usage of verb-based expressions drops down to 11.9%.

Root Tag	% in Dist. 2.1M Hashtags	% in All 60M Occurrences
Noun (N)	48.1	77.1
Verb (V)	26.0	11.9
INTJ (!)	5.8	3.4
Adjective (A)	2.5	1.3
Preposition (P)	2.0	1.0
Coord. Conjunction (CC)	1.0	0.4
Adverb (R)	0.4	0.3
Multi-root cases	12.5	3.5

Table 15: The most frequent root tags.

The third most common root tag is interjection (INTJ). However, we observe that many words tagged as interjection are actually unknown words not recognized by TweeboParser. The other single-root POS tag cases, namely adjective (A), preposition (P), coordinating conjunction (CC), and adverb (R) cover a very small portion. The rest of the cases covering 12.5% of distinct hashtags are complex expressions made of multi-root tags.

Analysis of Tag Patterns Around the Root

In order to inspect the structure of a hashtag, we investigate the tags that are connected to the root tag in the TweeboParser's output. We call these "tag patterns around the root tag." In the tables below, we show these patterns as a sequence of tags in the order of their occurrence. We also include the root tag, which is surrounded by brackets to make it distinguishable. In Figure 4 above, while the root tag is N, it has only one dependent tag, which is P. Hence, the tag pattern around the root tag is denoted as **[N] P**. Below, we analyze the most frequent tag patterns around each major root tag separately.

Tag Pattern	% in Dist.	% in All	Example Segmented Hashtag
N [N]	41.0	23.3	Lindas Piernas
[N]	11.3	59.1	Punicorn
A [N]	10.8	5.4	EXCELLENT GINA
N N [N]	7.1	1.3	Alex Volta Logo
[N] P	5.6	1.9	Definition Of Fail

Table 16: The most frequent tag patterns headed by a noun.

Table 16 lists the most frequent tag patterns headed by a noun root tag. As shown in the first row, 41% of distinct hashtags in noun form are constructed by combining two nouns together. It is almost four times more than the ones with single noun, consisting of 11.3% of cases. 10.8% of noun-headed hashtags have a single adjective modifier. When we consider the percentages in all occurrences, hashtags with single noun become dominant.

Tag Pattern	% in Dist.	% in All	Example Segmented Hashtag
[V] N	14.4	23.7	Unfollow diddy
N [V]	11.3	8.9	vegas sucks
N [V] N	6.4	4.0	no brain left today
[V] V	5.8	4.2	dont judge medotcom
O [V] N	4.7	3.9	this is genius

Table 17: The most frequent patterns headed by a verb.

Table 17 lists the most frequent patterns headed by a verb tag. This time the most frequent pattern is sentence in imperative form, which covers 14.4% of distinct verb-based hashtags. Note that the fourth case is also in imperative form, which

increases the total to 20.2%. The other three cases can be considered as regular sentence formations, whose percentage sums up to 22.4%. When looking at all occurrences, we observe that people tend to use imperative form more often than regular form.

Tag Pattern	% in Dist.	% in All	Example Segmented Hashtag
N [A]	29.1	9.2	cambio social
[A] P	15.2	3.1	scared of iPhone
R [A]	13.1	3.9	very confused
[A]	11.2	74.6	Willing
A [A]	6.2	1.3	deep undercover

Table 18: The most frequent patterns headed by an adjective.

Considering trapped sentiment inside hashtags, another important root tag to investigate is adjective. According to Table 18, surprisingly, 29.1% of adjective-headed hashtags include the adjective as a post modifier following a noun. On the other hand, 74.6% of all adjective-headed hashtags consist of single word adjective. However, when we consider all hashtags, not only the ones headed by adjectives, adjectives are mostly used inside expressions especially as a noun modifiers (**A [N]** in Table 16). This also supports our finding that sentiment is trapped inside multi-word hashtags considering the fact that adjectives carry most of the sentiment compared to other word types.

5. Conclusion

In this study, we explore extending a feature-based machine learning approach with context-based features and LM-based approaches for hashtag segmentation. We observe that context-based features improve the results without needing thousands of tweets in the context. As few as 20 tweets are sufficient. While using LM alone does not improve the results, utilizing N best segmentations given by LM as features helps us obtain the state-of-the-art results on both test sets. Moreover, as we add context-based features on top of that, we see up to 3 point increase when training is done on the HASHTAGS set, which makes HASHTAGS the best training set on Test-BOUN. Error analysis shows that most of the incorrectly segmented hashtags are due to low frequency words like foreign words, expressions with numbers, and special capitalized cases. In this study, we used the word boundaries in tweets to create the Tw-BOUN and Tw-STAN training data sets. One possible future work would be using segmented tweets (Li et al., 2015) to create training data, since tweet segments can be seen as phrases with similar characteristics to hashtags and may be more successful at representing word boundaries in hashtags compared to whole tweets.

In the second part of our research, we segment millions of hashtags extracted from the SNAP Tweet data set and then analyze their structure. We observe that almost 90% of 2.1M distinct hashtags include multiple words. Moreover, we observe that in 80% of sentiment bearing hashtags, sentiment is trapped inside multi-word hashtags. As we further analyze the parses of auto-segmented hashtags, we discover that about quarter of distinct hashtags are written as verb headed expressions, especially in imperative form. However, as we consider all occurrences, people tend to use noun-based hashtags more often. Adjectives are mostly used inside expressions rather than as single word hashtags. All these show that hashtags are not simple one-word labels, hence hashtag segmentation is necessary for a better understanding and utilization of hashtags, especially in the sentiment analysis task.

6. Acknowledgements

This research is supported by Boğaziçi University Research Fund Grant Number 11170. Arda Çelebi is also supported by ASELSAN Graduate Scholarship for Turkish Academicians and Arzucan Özgür is supported by the BAGEP Award of the Science Academy.

7. References

Bansal, Piyush, Bansal, Romil, and Varma, Vasudeva. (2015a). Towards deep semantic analysis of hashtags. *To Appear in 37th European Conference on Information Retrieval*, pages 453–464.

Bansal, Piyush, Jain, Somay, and Varma, Vasudeva. (2015b). Towards semantic retrieval of hashtags in microblogs. *In the proceedings of the 24th International Conference on World Wide Web Companion*, pages 7–8.

Berardi, Giacomo, Esuli, Andrea, Marcheggiani, Diego, and Sebastiani, Fabrizio. (2011). Isti@trec microblog track 2011: Exploring the use of hashtag segmentation and text quality ranking. *In the Proceedings of Twentieth Text REtrieval Conference*.

Billal, Belainine, Fonseca, Alexsandro, and Sadat, Fatiha. (2016). Named entity recognition and hashtag decomposition to improve the classification of tweets. *Proceedings of the 2nd Workshop on Noisy User-generated Text*.

Celebi, Arda and Ozgur, Arzucan. (2016). Segmenting hashtags using automatically created training data. *Proceedings of Language Evaluation and Resources Conference (LREC)*.

Chen, Songjian, Xu, Yabo, and Chang, Huiyou. (2012). A simple and effective unsupervised word segmentation approach. *In the Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence*.

Chen, Tao, He, Xiangnan, and Kan, Min-Yen. (2016). Context-aware image tweet modelling and recommendation. *Proceedings of the 2016 ACM on Multimedia Conference*.

Cunningham, Hamish, Maynard, Diana, Bontcheva, Kalina, and Tablan, Valentin. (2002). Gate: an architecture for development of robust hlt applications. *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics, pp 168175, Stroudsburg, PA, USA*.

Daelemans, Walter, van den Bosch, Antal, and Weijters, Ton. (1997). Igtree: Using trees for compression and classification in lazy learning algorithms. *Artificial Intelligence Review, 11*, pages 407–423.

Declerck, Thierry and Lendvai, Piroska. (2015). Processing and normalizing hashtags. *Proceedings of Recent Advances in Natural Language Processing (RANLP)*.

Jr., G. David Forney. (1973). The viterbi algorithm. *In the Proceedings of the IEEE, 61(3)*, pages 268–278.

Kong, Lingpeng, Schneider, Nathan, Swayamdipta, Swabha, Bhatia, Archna, Dye, Chris, and Smith, Noah A. (2014). A dependency parser for tweets. *In Proceedings of Empirical Methods on Natural Language Processing (EMNLP) 2014*.

Lee, Young-Suk, Papineni, Kishore, and Roukos, Salim. (2003). Language model based arabic word segmentation. *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics, pp 399-406. Sapporo, Japan*.

Li, Chenliang, Sun, Aixin, Weng, Jianshu, and He, Qi. (2015). Tweet segmentation and its application to named entity recognition. *IEEE Transactions on Knowledge and Data Engineering*, 2.

Maynard, Diana and Greenwood, Mark A. (2014). Who cares about sarcastic tweets? investigating the impact of sarcasm on sentiment analysis. *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC), Reykjavik, Iceland*.

Mikolov, Tomas, Sutskever, Ilya, Chen, Kai, Corrado, Greg, and Dean, Jeffrey. (2013). Distributed representations of words and phrases and their compositionality. *Proceedings of Neural Information Processing Systems (NIPS)*.

Nielsen, Finn Arup. (2011). A new anew: Evaluation of a word list for sentiment analysis in microblogs. *Proceedings of the ESWC2011 Workshop on Making Sense of Microposts: Big things come in small packages*.

Owoputi, Olutobi, O'Connor, Brendan, Dyer, Chris, Gimpel, Kevin, and Schneider, Nathan. (2012). Part-of-speech tagging for twitter: Word clusters and other advances. (CMU-ML-12-107).

Peng, Fuchun, Feng, Fangfang, and McCallum, Andrew. (2004). Chinese segmentation and new word detection using conditional random fields. *Proceedings of the 20th international conference on Computational Linguistics. Article 562*.

Qadir, Ashequl and Riloff, Ellen. (2014). Learning emotion indicators from tweets: Hashtags, hashtag patterns, and phrases. *In the proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*.

Rumelhart, D.E. and McClelland, J. L. (1986). On learning the past tenses of english verbs. *Parallel distributed processing: explorations in the microstructure of cognition, Vol. 2, MIT Press Cambridge, MA, USA*, pages 216–271.

Srinivasan, Sriram, Bhattacharya, Sourangshu, and Chakraborty, Rudrasis. (2012). Segmenting web-domains and hashtags using length specific models. *In the Proceeding CIKM 2012 Proceedings of the 21st ACM international conference on Information and knowledge, ACM New York, NY, USA*, pages 1113–1122.

Stilo, Giovanni and Velardi, Paola. (2017). Hashtag sense clustering based on temporal similarity. *Computational Linguistics*, pages 181–200.

Wang, Kuansan, Thrasher, Christopher, and Hsu, Bo-June. (2011). Web scale nlp: A case study on url word breaking. *In The International World Wide Web Conference*, pages 357–366.

Wong, Pakkwong and Chan, Chorkin. (1996). Chinese word segmentation based on maximum matching and word binding force. *Proceedings of the 16th conference on Computational linguistics*.

Xue, Nianwen. (2003). Chinese word segmentation as character tagging. *International Journal of Computational Linguistics and Chinese Language Processing volume 8(1)*.

Zhang, Yue and Clark, Stephen. (2008). Chinese segmentation with a word-based perceptron algorithm. *Annual Meeting of the Association of Computational Linguistics*, pages 888–896.