Segmentation of Tweets with URLs and its Applications to Sentiment Analysis

Abdullah Aljebreen, [1] **Weiyi Meng,** [2] **Eduard Dragut,** [1]
[1] Temple University
[2] Binghamton University
aaljebreen@temple.edu, meng@binghamton.edu, edragut@temple.edu

Abstract

An important means for disseminating information in social media platforms is by including URLs that point to external sources in user posts. In Twitter, we estimate that about 21% of the daily stream of English-language tweets contain URLs. We notice that NLP tools make little attempt at understanding the relationship between the content of the URL and the text surrounding it in a tweet. In this work, we study the structure of tweets with URLs relative to the content of the Web documents pointed to by the URLs. We identify several segment classes that may appear in a tweet with URLs, such as the title of a Web page and the user's original content. Our goals in this paper are: introduce, define, and analyze the segmentation problem of tweets with URLs, develop an effective algorithm to solve it, and show that our solution can benefit sentiment analysis on Twitter. We also show that the problem is an instance of the block edit distance problem, and thus an NP-hard problem.

Introduction

Many applications built upon Twitter data require a robust understanding of user exchanged messages (aka tweets), such as named entity recognition (NER) (Li et al. 2015; Ritter et al. 2011; Schneider, Mukherjee, and Dragut 2018), event detection and summarization (Hughes and Palen 2009), and sentiment analysis (SA) (Gezici et al. 2013; Yang, Dragut, and Mukherjee 2020). The length restriction on tweets encourages users to employ unconventional writing techniques when creating tweets. Thus, tweet analysis remains challenging despite more than a decade long research effort (Li et al. 2015; Kong et al. 2014).

This paper presents, to our knowledge, the first attempt to understand the text structure in a tweet that contains URLs. We refer to these tweets as ***URLtweets*** throughout the paper. We show that about 1 in 5 English-language tweets contains a URL (see the section "Tweets with URLs"). Another study (Suh et al. 2010) shows that 56.7% of retweets are to URLtweets. Since URLtweets represent a significant portion of the overall Twitter stream traffic, we contend that this class of tweets deserves its own attention. In this work, we introduce *the problem of URLtweets segmentation*, analyze its complexity, give an algorithm to solve it, and show that the proposed segmentation can improve sentiment analysis on tweets.

Tweet (1) [Tu]	**Bloomberg rips Trump: 'Totally incompetent'** https://t.co/...
Tweet (2) [TBu]	**Bloomberg rip #Trump: '#Totallyincompetent'.** ...New York billionaires, #Michael-Bloomberg ... https://t.co/...
Tweet (3) [TBUu]	**Bloomberg rips Trump: 'Totally incompetent'** ...This is a person who should not be the #President ... *We're paying the price for* ... https://t.co/...

Figure 1: Example of URLtweets, showing segmentation patterns (in square brackets on the left) and text segments marked as follows: boldface for title, underline for body and *italic* for user

First, we aim to show that the text of such tweets can be naturally partitioned into segments, substrings, that relate to the content of the web page pointed to by the URL, which we will refer to as ***URLdoc***. We define three classes of tweet segments: *title segment*, denoted by T (this segment overlaps with a large portion of the title of the URLdoc); *body segment*, denoted by B (the segment is an excerpt from the body of the URLdoc); and *user segment*, denoted by U (the segment is the user's utterance). We denote the presence of URL in a tweet by u. Figure 1 shows several examples of URLtweets with their segments and patterns. In Tweet (1), for instance, the entire text of the tweet, "Bloomberg rips Trump: 'Totally incompetent' ", is the title of the URLdoc. In Tweet (2), the title is followed by an excerpt from the body of the URLdoc. In Tweet (3), there are both title and body segments, and a user's original utterance, "We're paying the price for ...".

There are several challenges in the tweet segmentation problem that we will describe later in the paper. One challenge is when substrings of the title or the body are replaced with active Twitter-specific markups like in Tweet (2) (Figure 1). In that tweet "Trump" and "Totally Incompetent" are replaced by #Trump and #Totallyincompetent, respectively.

Title: Comcast raises bid for Sky to $34 billion, tops Fox's offer

Tweet (4) [Tu] **Comcast raises bid for Sky to $34 billion, tops Fox's offer** https://t.co/...

Tweet (5) [Tu] **#Comcast raises bid for #Sky to $34 billion, tops #Fox's offer** https://t.co/...

Tweet (6) [Tu] **Comcast raises bid for Sky to $34 billion** http://t.co/...

Title: 'Knives are too sharp and filing them down is solution to soaring violent crime', judge says

Tweet (7) [Tu] **Judge Says 'Knives are too sharp and filing them down is solution to soaring violent crime'** https://t.co/...

Figure 2: Examples of two sets of URLtweets, with the original title of the URLdoc shown before each set.

The above examples illustrate the kind of edits users may perform on URLtweets and suggest that the segmentation problem is an instance of the approximate string matching problem. This is traditionally solved with variants of the edit distance problem (Levenshtein 1966). However, many user edits are *block operations*, in which a whole substring of the input is edited. For example, Tweet (7) in Figure 2 shows an example where the substring "judge says" in the title is moved from the end of the title to the beginning of the title in the tweet (perhaps, the user seeks to change the tone of the message). In the section "Problem Analysis", we give a thorough analysis of the URLtweet segmentation problem by mapping it to various instances of the string block edit distance problem (SBED) and show that the problem is NP-hard when treated as SBED problem (Shapira and Storer 2002).

Second, we demonstrate that our segmentation procedure benefits sentiment analysis on tweets. Our belief is that the value of Twitter text mining lies in the users' original utterances. Hence, it is important to separate a user's own thoughts in a tweet from the pieces of text that are copied from Web documents. Without such a proper annotation of the ownership of the pieces of text in a tweet, the outcome of some NLP tasks may be difficult to interpret or even wrongly interpreted. We explore the benefit of segmentation to sentiment analysis in the sections "Sentiment Analysis" and "Case Study: Sentiment Analysis".

Motivation

In this section, we give quantitative evidence about the need to treat URLtweets as first-class citizens in NLP applications on Twitter. We first show that they represent an important portion of the Twitter traffic. Then, we show the potential improvement gain of sentiment analysis tools when the segmentation proposed in this paper is performed prior to their execution.

Tweet (8) [TuU] **He needed a kidney. A classmate from 50 years ago whom he barely knew ...** https://t.co/... *the way love is.*
Unsegmented: Negative **User's segment:** Positive

Tweet (9) [BUu] They're making a mistake because I have a gut, and my gut. . . *(Guess who!) Read the rest of the story* . . . https://t.co/...
Unsegmented: Negative **User's segment:** Neutral

Figure 3: Example of two URLtweets with their polarities, showing the erroneous polarities when ignoring segments. The markups are the same as in Figure 1.

Tweets with URLs

Our goal here is to estimate the proportion of tweets with URLs in the Twitter stream, which we refer to as p_u. We relied on our own crawling instead of estimations by previous research because we needed a more recent estimation using a large sample that is uniformly distributed over a time range. We conduct two studies toward this goal: (YM) a 12-month long sampling (Jan 2018 - Jan 2019) and (MD) 1-month long sampling (Nov 2018). In both studies, we used Twitter4J[1] to collect a random sample of the general stream of English-Language tweets without specifying any keywords, topics, or geographic regions. In YM, we crawled 400k tweets twice every month, for a total of 9.6 million tweets, and found that p_u lies between 0.205 to 0.206 with 95% confidence. In the MD study, we crawled 200k tweets daily for a month, for a total of 6 million tweets. This study gives an estimate of p_u in the interval 0.204 to 0.205 with 95% confidence, which shows that the outcome of the YM study holds when changing the sampling frequency. We thus infer that about 100M of the 500M daily tweets[2] are URLtweets. Hence, we contend that URLtweets require their own suite of NLP tools. This work is an effort, the first to our knowledge, toward this goal.

Sentiment Analysis

SA is the NLP task most often applied to Twitter streams (Gezici et al. 2013; Vanzo, Croce, and Basili 2014). Knowing which parts of a tweet are copied from an external source and which parts express the original opinion of the user is critical in opinion mining. Tweet segmentation as proposed in this work is beneficial to SA for at least two reasons. First, a large fraction of URLtweets, about 40% in our dataset (see the section "Experiments"), does not contain a user segment. Hence, one may choose not to run them through an SA tool. Second, existing SA tools may predict misleading polarities for the URLtweets that contain other segments besides user segments. We explain the reason below using the two tweets in Figure 3. Consider Tweet (8), which has the title of the article, the URL, and the user's own utterance. We tested two SA tools (StanfordCoreNLP, BERT) (Socher et al. 2013;

[1] http://twitter4j.org

[2] https://www.internetlivestats.com/twitter-statistics/

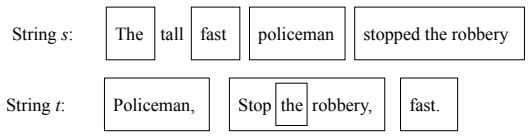

Figure 4: Example of substring families in two strings.

Müller, Salathé, and Kummervold 2020) on it and they both give a negative polarity for the text of this tweet. However, when we input only the user utterance, that we extract manually, they both give the correct user's sentiment: positive. Tweet (9) in Figure 3 has the pattern: body, user utterance, and URL. Again, the SA tools wrongly produce negative polarity for this tweet, whereas the user does not overtly express an opinion. If we input the correct user segment to the tools they both give neutral. In both cases existing tools are mislead by the text of the URLdoc, title or body, that does not belong to the user. We give a large scale experimental study in the section "Case Study: Sentiment Analysis".

Problem Definition

Background

We begin with some basic definitions to aid introducing the problem. A *string* is a sequence of elements, *letters*, from a finite set, *alphabet*, denoted by Σ. Given strings: s, z, x, w and v, we say the following: s is denoted by $s = s_1 s_2 \dots s_k$, where each s_i is a letter; $|s| = k$ is the length of s; s is called *empty* string when $|s| = 0$; s is a *substring* of x when it satisfies $x = wsv$, and when w or v are *empty*, s is also a *prefix* or *suffix* of x, respectively. Finally, we define a *k-block substring family* of a string s as a set $F_k(s)$ containing k substrings of s. If the substrings in $F_k(s)$ do not overlap, we say the family is *disjoint*. If each character of s is contained in some substring, we say the family represents a *cover* of s. For example, consider the strings s and t in Figure 4, each with their substring family annotated using the boxes. The substring family of t is a *cover* and not *disjoint*, while that of s is not a *cover* and it is *disjoint*.

Edit Operations: We have stated that a Twitter user may perform a number of operations on excerpts from URLdoc before posting them in their tweet. These operations include the usual operations used to define traditional *edit distance* – insert letter (`ins`) and delete letter (`del`): the former turns a string $s = s^1 s^2$ to $s^1 a s^2$ and the latter $s' = s^1 a s^2$ to s for any letter $a \in \Sigma$ and strings s^1, s^2. The replace operation of a single letter with another can be simulated by insert and delete. For example, in Figure 2 we show in Tweet (4) a tweet that makes an exact copy of the title of a document. In Tweet (5), the user makes several insert edits to the title, e.g., changing "Comcast" to "#Comcast". In Tweet (2) (Figure 1), the user performs both an insert and a delete on "Totally incompetent" to transform it into "#Totallyincompetent".

Users also tend to perform bulk operations, where an entire substring (*block*) is edited at once. There are several block operations defined in the literature, such as block insert (`b-ins`), block delete (`b-del`) and block move (`b-mv`), which transform a string $s = s^1 s^2 s^3$ to $s^1 s^2 s^3 s^4$, $s^1 s^3$ and $s^3 s^2 s^1$, respectively (Ganczorz et al. 2018). We exemplify `b-del` and `b-mv` on the example tweets in Figure 2. In Tweet (6) the user performs a block delete to remove the tail of the title. In Tweet (7), we show an example of a block move ("Judge Says"), from the end of the title to the beginning. We adhere to the notation in (Ganczorz et al. 2018) and denote by $ED_{Op}(s,t)$ the minimal number of operations, from the set of allowed edits Op, that transform string s to string t.

Definitions

The input to our problem is a URLtweet with one or more URLs in addition to regular text. We describe such tweets with the pattern $t^{0|1}(ut^{0|1})^m$, $m > 0$, where u stands for the occurrence of a URL and t for the occurrence of a piece of text. In our dataset, the two most dominant patterns are tu and tut, with 66% and 21% occurrences, respectively. Only 5% of URLtweets contain more than one URL. Hence, without loss of generality, and for ease of presentation, we assume that each URLtweet has only one URL. We use α and d_u to refer to URLtweets and their associated URLdocs, respectively. We seek to solve the following problem:

User Utterance Problem (UUP): Find all the substrings in α that represent a user's utterance.

We say substrings because the user's own content in a tweet may not be one continuous sequence of characters in the tweet, but rather interleaved with other pieces of text that are excerpts from the URLdoc. It is difficult to separate the substrings that pertain to a user's own thoughts from the rest in a tweet. It is easier to detect the parts in the tweet that originate from the URLdoc.

Let s be a substring of t, where t is defined as above. We label s as a URLdoc segment if: (i) s is a (non-trivial) substring of URLdoc and (ii) any other substrings: xs, sx, or xsx of t are not substrings of URLdoc ($x \in \Sigma$). The complement problem of UUP is:

Complement UUP ($\overline{\text{UUP}}$): Find all URLdoc segments in α.

If we are able to solve $\overline{\text{UUP}}$, then we can obtain a user's own utterance (if any) by removing those URLdoc segments from α. There are two ways to look at the problem: (1) from URLtweet to URLdoc, and (2) from URLdoc to URLtweet. The latter gives a more intuitive view of the problem as one can imagine that the user constructs the URLtweet from the URLdoc by discarding large pieces (entire paragraphs) of the URLdoc, keeping parts of the URLdoc (as illustrated in Figure 5), and adding her own words.

The classic edit distance model – where a comparison between two given strings, s and t, is performed by computing the cost of the character-level operations needed to transform string s into t (Ann et al. 2010) – is not suitable to model URLdoc to URLtweet user editing. The issue is that this model disregards the structure of d_u into phrases and paragraphs. For example, consider Tweet (7) in Figure 2 where the only difference between its copy of the title and the original title is the movement of the last two words to the beginning. With the traditional edit distance, the cost of transforming it into the title gives a cost of 10 operations

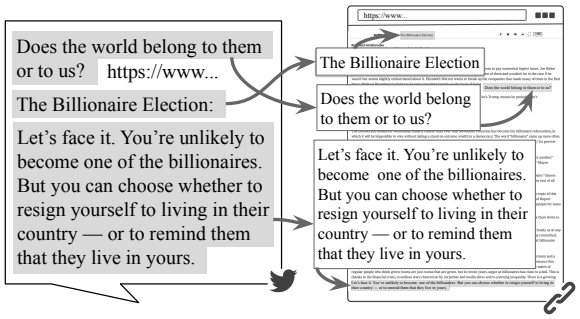

Figure 5: Example of tweet with a URL (left) and a snapshot of the corresponding web document (right). Notice the 3 annotated segments and their positions in the document.

which is too high and not reflective of the similarity between the two strings. The cost is only one operation when we use blocks of substrings since the only required operation is one block move and that captures the high-level structure of the text (Lopresti and Tomkins 1997). We argue that this model is a more appropriate computational model for our problem.

Let us define the goal of $\overline{\text{UUP}}$ more formally given the previous constraints. Our goal is to choose the substring families $F_k(d_u) = \{d_u^1, ..., d_u^k\}$, $F_k(\alpha) = \{\alpha^1, ..., \alpha^k\}$ and a function $m : F_k(d_u) \longleftrightarrow F_k(\alpha)$ such that $m(d_u^i) = \alpha^j, i, j \in [1..k]$, and d_u^i and α^j are not *both* empty strings. We say that

1. if $|d_u^i| > 0$ and $|\alpha^j| > 0$ then blocks d_u^i and α^j *match*.
2. if $|\alpha^j| = 0$ then user "deletes" block d_u^i.
3. if $|d_u^i| = 0$ then user "inserts" block α^j.

There are three aspects we need to discuss further in the formal definition of $\overline{\text{UUP}}$: (1) matching, (2) the k that gives the desired outcome, and (3) the overlap/disjoint properties of the k-block substring families $F_k(d_u)$ and $F_k(\alpha)$.

Matching

The above formulation of $\overline{\text{UUP}}$ assumes that a user performs *exact copies* from a URLdoc to a URLtweet. This helps give an intuitive understanding of our goals in this work. As discussed earlier in the paper and exemplified in Tweets (2) and (3) in Figure 1, users may choose to edit the copied text from URLdoc before posting the URLtweet. We need to treat the detection of URLdoc segments in a tweet as a problem of approximate string matching. Hence, we require an underlying distance function that returns the cost of aligning a substring of d_u with a substring of α. In practice, one may use the traditional string edit distance. However, our approach and its complexity do not change, when using other functions.

Optimization Goal

There are exponentially many k-block families for both d_u and α. The questions are which one of them gives the desired outcome and how to find it. Intuitively, our goal is two-fold: (1) to cover as much of the content of URLTweet as possible with substrings from URLdoc (i.e., detect the copies in URLtweet that are from URLdoc) and (2) for each matching pair of block substrings, the distance between the two should be small. Therefore, we seek to find the k-block families for both d_u and α with the following property:

$$\min_k \min_{F_k(d_u), F_k(\alpha)} \min_m \left\{ k \cdot c_{block} + \sum_{i=1}^{k} dist(d_u^i, m(d_u^i)) \right\}$$

We can assume a unit cost for the block operations in the above equation, c_{block}, without loss of generality in our problem. This is the model followed by the general string block-edit distance (SBED). SBED includes other block moves that we do not see applicable in our settings such as block uncopy and block reversal (Muthukrishnan and Sahinalp 2000). The SBED model is flexible enough and can capture similarity both at low-level (characters) and high-level (blocks), by allowing matched blocks to be further edited with character operations. There are many cases of the problem in general (Ganczorz et al. 2018; Lopresti and Tomkins 1997; Shapira and Storer 2002), some of which have tractable solutions, while others are hard and require approximate algorithms. We will analyze them in the next section to determine which one of them fits $\overline{\text{UUP}}$.

Problem Analysis

Cover: We consider first the substring family of URLtweet. As illustrated in Figures 1 and 2, besides copying content from a URLdoc, a user may include original content in the URLtweet. Hence, $F_k(\alpha)$ need not be a cover since we do not require that every substring of $F_k(\alpha)$ has a correspondent in $F_k(d_u)$. On the other hand, $F_k(d_u)$ can be either a cover and not a cover, depending on whether we seek to match the title or the body, respectively.

Disjoint: We assert that the substring families of URLtweets and URLdoc have to be disjoint. Allowing the blocks of a URLtweet to overlap when matching them to those of a URLdoc is counter-intuitive since each piece of copied text originates from a single location in the URLdoc. And, vice-versa, allowing blocks of a URLdoc to overlap corresponds to a user having multiple copies of a piece of text from URLdoc in her URLtweet (i.e., she has redundant text in her tweet). When tweets contain such "redundancy," it is our observation that the redundant parts originate in segments of different classes, for example, a nontrivial portion of the title is repeated in the body.

We conclude that among the multiple SBED models (Lopresti and Tomkins 1997), only two are viable models to capture the URLtweet segmentation problem, namely $\bar{C}D$-CD and $\bar{C}D$-$\bar{C}D$. The first one is more suitable when discovering title segments while the second one is more suitable to model body segments of a URLtweet. Both variations of the problem are NP-complete (Lopresti and Tomkins 1997). Hence, we state that:

Lemma 1 *$\overline{UUP}$ is NP-complete.*

A number of polynomial approximation solutions have been proposed in the literature (Cormode and Muthukrishnan 2007; Ganczorz et al. 2018), the most recent of which is in (Ganczorz et al. 2018). We build our algorithm on the approximation greedy algorithm in (Shapira and Storer

Algorithm 1: The pseudo code of the main function of our algorithm.

match(d_u, α)
  `label` $lcs(d_u, \alpha)$ as lcs_{block}
  d_u^p, α^p= the prefixes of lcs_{block} in d_u and α
  `match` (d_u^p, α^p)
  d_u^s, α^s= the suffixes of lcs_{block} in d_u and α
  `match` (d_u^s, α^s)
  `trim` non-lcs_{block} prefixes/suffixes in α
return d_u, α
check(d_u, α)
  if $similarity(d_u, \alpha) \geq threshold$ **then**
    return α as a segment of d_u
  else
    d_u, α=`reduce` (d_u, α)
    `match` (d_u, α)
  end

2002). This has a number of desirable properties, including that when only block deletion and block copy are allowed, this gives an $\mathcal{O}(1)$ approximation of the distance. Although, our problem is hard in general, most of the encountered instances are not as complex as those treated by SBED, e.g., computational biology.

Segmentation Method

In this section, we present an efficient segmentation algorithm that produces the required segments of URLtweets in a polynomial run-time. In this segmentation method, we expect two text inputs: a URLtweet (a tweet with one URL along with pieces of text) and a URLdoc (the web document corresponding to the URL in the URLtweet, which we retrieve beforehand). Before the segmentation step, we normalize the text of the URLtweets and URLdocs, e.g., removing the URL from the URLtweet, trimming extra spaces and newline characters, and omitting special characters except for the *pound #* and *at @* symbols, because of their special meanings in tweets and their importance to the segmentation process. We note that emojis and other characters we removed in this cleaning stage may be important signals for some downstream NLP tasks. Hence, once each segment's boundaries are detected, we restore the content of each segment to its original content as it appeared in the raw tweet.

The Greedy Algorithm

Our algorithm commences with the text of the URLtweet, α, and the URLdoc d_u, which can be either the title or the body, and attempt to extract the correct segments within α. Finding body segments is more expensive than finding title segments in URLtweet. In the worst case, we need to scan an entire (long) Web document when looking for a body segment. Therefore, it is computationally beneficial to look for the title segment first, remove it from the URLtweet, and then scan the URLtweet minus the title segments for the presence of body segments B. We label the remaining segments, if any, as user utterance.

Tweet a)	Fired officer is charged with third degree murder after George Floyd's death. #BlackLivesMatter #GeorgeFloyd: Absolute Chaos in **#Minneapolis as Protests Grow Across**[1] #US The New York Times
Title	George Floyd Updates Absolute Chaos in **Minneapolis as Protests Grow Across**[1] US The New York Times

Tweet b)	**George Floyd**[1]s death. #BlackLivesMatter #GeorgeFloyd: **Absolute Chaos in**[2] **#Minneapolis as Protests Grow Across**[3] **#US The New York Times**[4]
Title	**George Floyd**[1] Updates **Absolute Chaos in**[2] **Minneapolis as Protests Grow Across**[3] **US The New York Times**[4]

Tweet c)	s death. #BlackLivesMatter #GeorgeFloyd: **Absolute Chaos in**[1] **#Minneapolis as Protests Grow Across**[2] **#US The New York Times**[3]
Title	George Floyd Updates **Absolute Chaos in**[1] **Minneapolis as Protests Grow Across**[2] **US The New York Times**[3]

Tweet d)	**George**[1] **Floyd**[2]: **Absolute Chaos in**[3] **#Minneapolis as Protests Grow Across**[4] **#US The New York Times**[5]
Title	**George**[1] **Floyd**[2] Updates **Absolute Chaos in**[3] **Minneapolis as Protests Grow Across**[4] **US The New York Times**[5]

Figure 6: An example of applying Algorithm 1 as discussed in the section "The Greedy Algorithm"

Our algorithm relies upon finding the longest common substrings (LCS) between α and d_u to detect the desired URLdoc segments. Utilizing LCS is both effective and efficient in approximate string matching (Shapira and Storer 2002). Their algorithm reduces the two given strings to shorter strings by replacing repeatedly a longest common substring by a new single character. The traditional edit distance is then applied on the new strings (Shapira and Storer 2011). We use this technique (i.e. the repeated tagging of LCS between two strings) in our proposed algorithm and add a few heuristics to help the matching process in coping with character usage specific to Twitter, such as the use of hashtags *pound #* and at-user *at @*.

The pseudocode in Algorithm 1 depicts the main procedure of our approach which consists of two steps: $match$ and $check$. The goal is to obtain a single d_u-segment from an α, assuming that d_u is the title in this case. The first portion of the algorithm is the *match* function on which we employ a solution for the Longest Common Substring problem (LCS) to construct a candidate segment. In the $check$ part, we check if the candidate is acceptable, according to our measures. If not, we reconfigure its limits, trying to improve

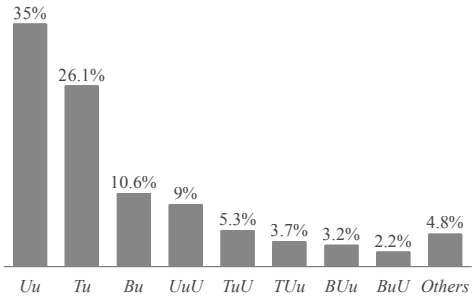

Figure 7: Summary of the most common tweet patterns

its quality. We use a running example (Figure 6) to describe these two steps in more detail.

In line 2, we look for the longest common substring between α and d_u and mark it as a single lcs_{block}, which we highlight in bold in Figure 6-a. Next, in lines 3 and 5, we refer to the remaining parts of the texts (before and after lcs_{block}) as prefixes and suffixes. In lines 4 and 6, we recursively look for additional lcs_{blocks} between the remaining prefixes and suffixes. The recursive calls continue until we find all possible lcs_{blocks} (Figure 6-b). Next, we trim α by removing substrings before and after the first and last discovered lcs_{blocks}. (line 7, Figure 6-b).

In the second part of the algorithm, we check the validity of the candidate segment, α, using our similarity measure (lines 10-11) and if it is valid, we return it. If not, we proceed to modify α using $reduce$. In $reduce$, we check the first and last lcs_{blocks} and remove the shortest one (line 13, Figure 6-c). Then, we call $match$ on the reduced α (line 14). In this new call, we look for more lcs_{blocks} in the trailing substring of α (Underlined substrings in Figure 6-c). We continue the iteration *match-check/reduce* until either we find a valid segment (Figure 6-d) or end up with strings that are too short to result in meaningful segments.

The polynomial computational complexity of our proposed algorithm follows from the algorithm introduced by Shapira and Storer 2002. Our heuristics to cope with characters specific to tweets do not add to the its running time.

Lemma 2 *The running time of Algorithm 1 is* $O(|d_u| \cdot |\alpha|)$

Experiments

We present an extensive empirical evaluation of our approach in this section to show (i) the effectiveness of our segmentation algorithm and (ii) its benefit to sentiment analysis on tweets.

Data

We use the data collected in the YM study – the year-long crawl. It has 9.6M tweets. Our collection of tweets is randomly sampled from English-language tweets and not associated with any specific topic or region. We keep only the URLtweets from this data set, yielding about 2M tweets. The final data set contains about 1M tweets after further filtering. We filter out the following tweets: (i) tweets with no text (just URLs), (ii) tweets with very short pieces of text – less than three words, (iii) tweets with inaccessible URLs, and (iv) tweets with URLs that point to web pages with only media content or very short main text (aka body text).

Segmentation Patterns

We report the discovered patterns according to the output of our algorithm on the entire set of 1M tweets. We observe 82 different patterns. The top-8 most frequent patterns (Figure 7) cover 95.2% of all tweets. Notice that not all tweets include user original content U; only about 59.62% of them do. On the other hand 37.58% of the tweets contain title segments T and 17.61% have body segments B. Note that these percentages add up to more than 100% because some tweets are of multiple patterns, e.g., some tweets have both T (title) and U (user content). This observation can greatly help online NLP systems in practice because they may decide to omit tweets without user comment. This may also improve the overall sentiment analysis accuracy, which we show in the section "Case Study: Sentiment Analysis".

Segmentation Experiments

This is our main empirical study on segmentation. We seek to evaluate the performance of our entire segmentation approach. This evaluation is challenging since we do not have a set of ground truth labels. We conduct a "hard" evaluation of segmentation: if the algorithm labels incorrectly at least one of the three possible text segments for a tweet, the tweet is marked as incorrect. Given a large number of tweets we approximate the accuracy via sampling: with (i) simple random sampling and (ii) stratified random sampling. According to (Yilmaz, Kanoulas, and Aslam 2008), stratified sampling can unitize a better accuracy estimation from incomplete judgments. We draw 1,000 sample tweets in both cases, about 0.1% of the entire set of URLtweets in our dataset. The estimated accuracy is a sample proportion. With 95% confidence, we expect the accuracy, $\hat{\pi}$, to be in the interval [0.882, 0.919]. We manually investigate the 1,000 tweets and get $\hat{\pi} = 0.901$. In (ii), we use 9 classes corresponding to the top-8 patterns and others. The manual inspection gives the proportion estimate $\hat{\pi} = 0.898$, which falls into the confidence interval [0.879, 0.917] at 95% confidence level.

The stratified sample is useful. The results also show that patterns with B or both B and U segments are the most challenging patterns, such as Bu, BUu, and BuU, with estimated accuracies of 77.67%, 79.31% and 75.11%, respectively. We believe that the source of this issue is the similarity between B and U. In many cases, the user input can be similar to the content of the URL. On the other hand, Tu, TuU and Uu have the highest accuracies of 99.57%, 97.92% and 88.21%, respectively.

Case Study: Sentiment Analysis

We observe that most SA tools remove URLs from tweets before conducting SA. This blind removal indicates that SA tools lose any relationship between the pieces of text in URLtweets that belong to the user and those copied from URLdocs. Our goal in this section is to show via large scale studies the benefit of tweet segmentation in SA.

Tweet (10) [Bu]	Lady Gaga is clearly delighted at the gift of a grey horse, delivered on Monday to her Malibu, California, home... https://t.co/...

Tweet (11) [BUu]	"I didn't go round slagging off Tony Blair" - KL*but he did with Gordon Brown*:https://t.co/...

Figure 8: Two tweets from the SemEval-2017 dataset. Tweet (10) is positive and Tweet (11) is neutral.

Datasets. We run our SA experiments on two datasets: (1) *URLtweets-100k*: We select 100k tweets from our dataset presented in the section "Data". All of these tweets contain at least one *U* segment. (2) *SemEval* is a collection of 60k labeled tweets for the SemEval tasks (2013-2017) (Rosenthal, Farra, and Nakov 2017). It contains about 7k tweets with valid URLs, 2,523 of which have mixed segments (i.e. *U* segments and *T/B* segments).

SA Tools. We use the following Sentiment Analysis tools in this study: (1) *StanfordCoreNLP-SA*: It is part of the Stanford NLP suite (Socher et al. 2013). (2) *BERT-covid*: a transformer-based model, pre-trained on a large corpus of tweets on the COVID-19 topic. This model shows a noticeable performance gains even when working with general tweets compared to the standard BERT (Devlin et al. 2019; Müller, Salathé, and Kummervold 2020).

Study A. In this study, our goal is to look for signals of influence on the polarity of URLtweets when applying the segmentation. We use *URLtweets-100k* dataset. We compare the sentiment of the original (raw) tweets against that of the extracted user utterances using StanfordCoreNLP-SA tool (Socher et al. 2013). In this experiment, 79.39% of the original tweets receive positive/negative sentiment. But, only 29.71% of user segments receive positive/negative polarity, and that shows a large discrepancy between the intended user opinion and the opinion derived from the raw tweet.

Study B. In this study, we seek to show that the change in the URLtweets polarity after the segmentation is an improvement to the outcome of SA tools. Consider the following two tweets in Figure 8 from the *SemEval* dataset. Tweet (10) has polarity positive, although the user simply cites a passage from the article. In other words, Tweet (10) includes no user segment. So, we contend that it should be neutral. Tweet (11) has polarity neutral, but we believe that the user sentiment is negative.

We use the labeled *SemEval* dataset in this experiment and apply an SA tool before and after removing the non-user segments. The results show an increase in the F_1 score from 0.687 to 0.734 when using *BERT-covid* tool. We believe that the actual improvement may be higher, since the human labeled polarities are misguided by the mixed sentiments in some URLtweets as we showed earlier in this section.

Related Work

Text segmentation has different meanings across research communities. For instance, NLP text segmentation is the task of dividing a document into segments, such that each segment is a homogeneous text units which might include NLP tasks such as part-of-speech tagging, named entity recognition, and chunking (Hearst 1994; Kozima 1993; Utiyama and Isahara 2001; Lafferty, McCallum, and Pereira 2001; Zhang et al. 2019, 2018). In Data Mining and Database communities, text segmentation is the task of partitioning an input text (usually extracted from Web pages returned by search engines) into structured records of a target schema before loading them into a database (Agichtein and Ganti 2004; Borkar, Deshmukh, and Sarawagi 2001; Bing, Lam, and Gu 2011; Lu et al. 2013; Dong, Dragut, and Meng 2019). There is also the problem of segmenting scientific literature, which aims to identify and mark the underlying structure of scientific papers and abstracts (Hirohata et al. 2008), e.g., identify and classify sentences into sections such as Introduction, Method, Result, and Conclusion. Another example is a segmenting task for tweets that aims to partition a tweet into consecutive word n-grams (Li et al. 2015), where a segment may denote an entity mention or a "semantically meaningful unit".

In our segmentation task the segmentation of a tweet is *performed with respect to a document* which precludes an outright use of supervised techniques, including the deep neural network ones (Akhundov, Trautmann, and Groh 2018), because such methods require a large amount of labeled data (difficult to obtain in practice), and one still needs to compare the tweet content against that of the document.

Many works explore the tweet-specific features (at- mentions, hashtags, and URLs) in a variety of tasks. For example, they are employed in developing part-of-speech tagging for tweets (Gimpel et al. 2011), labeling emotions in tweets (Hasan, Agu, and Rundensteiner 2014), determining users' geo-locations (Compton, Jurgens, and Allen 2014), and predicting the popularity of a tweet (Suh et al. 2010; Jenders, Kasneci, and Naumann 2013). However, these efforts primarily make use of at-mentions and hashtags, and not of URLs. For example, hashtags are used primarily in topic related tasks to collect relevant tweets (Ray Chowdhury, Caragea, and Caragea 2019). We found that only few researchers recognize and exploit the advantage of tweets with URLs in mainstream tasks. For example, the work by Plank et. al. 2014 uses the rich linguistic context in URLs as a distant supervision in their POS tagging in Twitter.

Conclusions

In this paper, we introduced the problem of segmenting tweets with URLs with respect to the content of the documents pointed to by the URLs. We analyzed the complexity of the problem and showed that it is an NP-hard problem. We gave a greedy algorithm with a proven linear approximation of the ideal solution. We defined three types of segments: document title, document body, and user content. The proposed segmentation algorithm achieved an accuracy of about 90%. We presented a case study on SA of URLtweets and showed that omitting non-user text from tweets can improve sentiment analysis accuracy. Future work includes an additional study on the benefit of the proposed segmentation on other NLP applications.

Acknowledgments

This work was supported in part by the U.S. National Science Foundation BIGDATA 1546480 and 1546441 grants.

References

Agichtein, E.; and Ganti, V. 2004. Mining Reference Tables for Automatic Text Segmentation. In *SIGKDD*, 20–29.

Akhundov, A.; Trautmann, D.; and Groh, G. 2018. Sequence Labeling: A Practical Approach. *CoRR* abs/1808.03926.

Ann, H.-Y.; Yang, C.-B.; Peng, Y.-H.; and Liaw, B.-C. 2010. Efficient Algorithms for the Block Edit Problems. *Inf. Comput.* 208(3): 221–229.

Bing, L.; Lam, W.; and Gu, Y. 2011. Towards a Unified Solution: Data Record Region Detection and Segmentation. In *CIKM*, 1265–1274.

Borkar, V.; Deshmukh, K.; and Sarawagi, S. 2001. Automatic Segmentation of Text into Structured Records. In *SIGMOD*, 175–186.

Compton, R.; Jurgens, D.; and Allen, D. 2014. Geotagging One Hundred Million Twitter Accounts with Total Variation Minimization. In *IEEEBig Dat)*, 393–401.

Cormode, G.; and Muthukrishnan, S. 2007. The String Edit Distance Matching Problem with Moves. *ACM Trans. Algorithms* 3(1): 2:1–2:19.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL*, 4171–4186.

Dong, Y.; Dragut, E. C.; and Meng, W. 2019. Normalization of Duplicate Records from Multiple Sources. *IEEE Trans. Knowl. Data Eng.* 31(4): 769–782.

Ganczorz, M.; Gawrychowski, P.; Jez, A.; and Kociumaka, T. 2018. Edit Distance with Block Operations. In *ESA*, volume 112 of *LIPIcs*, 33:1–33:14.

Gezici, G.; Dehkharghani, R.; Yanikoglu, B.; Tapucu, D.; and Saygin, Y. 2013. SU-Sentilab : A Classification System for Sentiment Analysis in Twitter. In *SemEval*, 471–477.

Gimpel, K.; Schneider, N.; O'Connor, B.; Das, D.; Mills, D.; Eisenstein, J.; Heilman, M.; Yogatama, D.; Flanigan, J.; and Smith, N. A. 2011. Part-of-speech Tagging for Twitter: Annotation, Features, and Experiments. In *ACL*, HLT '11, 42–47.

Hasan, M.; Agu, E.; and Rundensteiner, E. 2014. Using hashtags as labels for supervised learning of emotions in twitter messages. In *ACM SIGKDD workshop on health informatics, New York, USA*.

Hearst, M. A. 1994. Multi-paragraph Segmentation of Expository Text. In *ACL*, 9–16.

Hirohata, K.; Okazaki, N.; Ananiadou, S.; and Ishizuka, M. 2008. Identifying Sections in Scientific Abstracts using Conditional Random Fields. In *IJCNLP*.

Hughes, A.; and Palen, L. 2009. Twitter Adoption and Use in Mass Convergence and Emergency Events. *International JOURNAL of Emergency Management* 6: 248–260.

Jenders, M.; Kasneci, G.; and Naumann, F. 2013. Analyzing and predicting viral tweets. In *WWW*, 657–664.

Kong, L.; Schneider, N.; Swayamdipta, S.; Bhatia, A.; Dyer, C.; and A. Smith, N. 2014. A Dependency Parser for Tweets. In *EMNLP*.

Kozima, H. 1993. Text Segmentation Based on Similarity Between Words. In *ACL*, 286–288.

Lafferty, J. D.; McCallum, A.; and Pereira, F. C. N. 2001. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *ICML*, 282–289.

Levenshtein, V. 1966. Binary Codes Capable of Correcting Deletions, Insertions and Reversals. *Soviet Physics Doklady* 10: 707.

Li, C.; Sun, A.; Weng, J.; and He, Q. 2015. Tweet Segmentation and Its Application to Named Entity Recognition. *IEEE Trans. on Knowl. and Data Eng.* 27(2): 558–570.

Lopresti, D. P.; and Tomkins, A. 1997. Block Edit Models for Approximate String Matching. *Theor. Comput. Sci.* 181: 159–179.

Lu, Y.; He, H.; Zhao, H.; Meng, W.; and Yu, C. 2013. Annotating Search Results from Web Databases. *IEEE Trans. on Knowl. and Data Eng.* 25(3): 514–527.

Müller, M.; Salathé, M.; and Kummervold, P. E. 2020. COVID-Twitter-BERT: A Natural Language Processing Model to Analyse COVID-19 Content on Twitter. *arXiv preprint arXiv:2005.07503* .

Muthukrishnan, S.; and Sahinalp, S. C. 2000. Approximate Nearest Neighbors and Sequence Comparison With Block Operations. In *STOC*, 416–424.

Plank, B.; Hovy, D.; McDonald, R.; and Søgaard, A. 2014. Adapting taggers to Twitter with not-so-distant supervision. In *COLING*, 1783–1792.

Ray Chowdhury, J.; Caragea, C.; and Caragea, D. 2019. Keyphrase Extraction from Disaster-Related Tweets. In *WWW*, 1555–1566. ISBN 9781450366748.

Ritter, A.; Clark, S.; Etzioni, O.; et al. 2011. Named entity recognition in tweets: an experimental study. In *EMNLP*, 1524–1534.

Rosenthal, S.; Farra, N.; and Nakov, P. 2017. SemEval-2017 Task 4: Sentiment Analysis in Twitter. In *Proceedings of SemEval '17*.

Schneider, A. T.; Mukherjee, A.; and Dragut, E. C. 2018. Leveraging Social Media Signals for Record Linkage. In *WWW (The Web Conference)*, 1195–1204.

Shapira, D.; and Storer, J. 2011. Edit Distance with Block Deletions. *Algorithms* 4. doi:10.3390/a4010040.

Shapira, D.; and Storer, J. A. 2002. Edit Distance with Move Operations. In *CPM*, 85–98.

Socher, R.; Perelygin, A.; Wu, J.; Chuang, J.; Manning, C. D.; Ng, A. Y.; and Potts, C. 2013. Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank. In *EMNLP*.

Suh, B.; Hong, L.; Pirolli, P.; and Chi, E. H. 2010. Want to Be Retweeted? Large Scale Analytics on Factors Impacting Retweet in Twitter Network. In *SCSM*, 177–184.

Utiyama, M.; and Isahara, H. 2001. A Statistical Model for Domain-Independent Text Segmentation. In *ACL*, 499–506.

Vanzo, A.; Croce, D.; and Basili, R. 2014. A context-based model for Sentiment Analysis in Twitter. In *COLING*, 2345–2354.

Yang, F.; Dragut, E.; and Mukherjee, A. 2020. Predicting Personal Opinion on Future Events with Fingerprints. In *COLING*, 1802–1807.

Yilmaz, E.; Kanoulas, E.; and Aslam, J. A. 2008. A Simple and Efficient Sampling Method for Estimating AP and NDCG. In *SIGIR*, 603–610.

Zhang, S.; He, L.; Dragut, E. C.; and Vucetic, S. 2019. How to Invest my Time: Lessons from Human-in-the-Loop Entity Extraction. In *SIGKDD*, 2305–2313.

Zhang, S.; He, L.; Vucetic, S.; and Dragut, E. 2018. Regular Expression Guided Entity Mention Mining from Noisy Web Data. In *EMNLP*, 1991–2000.