Hashtag Segmentation of Conversational Tweets in Turkish, Modelling Segmentation Approach Final Report

Utku Saridede, Sevket Topuz

Advisor: Ass. Prof. Arzucan Ozgur

Co-Advisor: Arda Celebi

Degree of Bachelor of Science

Natural Language Processing and Informational Retrival, Department of Computer Engineering, Bogazici University, Istanbul Bebek 34342, TR

Email: utku.saridede@boun.edu.tr, sevket.topuz@boun.edu.tr

Twitter is the latest social networking tool which affects everything related to the person. Twitter allows its users to write at most 140 character long update, it is known off as "tweet". In this research, analyzing segmentation of hashtags from tweets is our main objective. There are several researches in English, but not that much in Turkish. Studying with the Turkish corpus is somehow hard to study, because Turkish resources have grammer problems due to the English effect. The usage of the hashtags differ in country to country. Having more than one word in the hashtag or lapsus calami prevent researchers to work properly. It is the first project in Turkey to segment tweet's hashtags in Turkish. Therefore, the results of our project is milestone in that manner. It will assist oncoming projects in the case of corpus and method needs. The other part of the project is analyzing hashtags in the way of linguistics. Analyzed corpus will give more information about related countries. In the other words, short-term hashtag analyses keep informed about spesific situations which influence the society. After all said and done, we will recognize the strength of computer science on natural languages.

Keywords: Twitter; Tweet; Tweets; Hashtag; Segmentation; Turkish; Conversational Tweets; Hashtag Segmentation

Received 07 October 2015; revised 12 January 2016

CONTENTS

1. INTRODUCTION AND MOTIVATION

1.1. Introduction

1.1.1. The Story of Hashtag

A hashtag is a type of label or metadata letters used especially on social network and microblogging services which makes it easier for users to find messages with a specific theme or content.

The story is began with Twitter but has extended to other social media platforms as well as "facebook". In 2007, developer Chris Messina proposed, in a tweet, that Twitter begin grouping topics using the hash symbol. Twitter initially rejected the idea. But in October 2007, citizen journalists began using the hashtag "#SanDiegoFire", at Messinas suggestion, to tweet updates on a series of forest fires in San Diego.

1.1.2. Decision of Hashtags

Which characters can be defined as #hashtag is the important part of our project.

A hashtag is defined by any string prefixed with a "#", for instance, #freedomtomark, #shesuggest. The string can be a single word, an acronym, or multiple words joined together, and usually identifies the subject topic of the tweet (e.g., #ENG493) or expresses a comment about it (e.g., #kappamevku).

Spaces are an absolute segmentation rule. Even if hashtag contains multiple words, they should be together. Using capital letters in between words have no meaning. (#CahitArf). Uppercase letters will not alter search results, so searching for #CahitArf will yield the same results as #cahitarf.

Numbers are supported in Twiter, so as #23NisanBayrami. However; punctuation marks, commas, periods, exclamation points, question marks and apostrophes are forbidden characters. In addition to them; asterisks, ampersands or any other special characters are also restricted ones. There is no preset list of hashtags. Creating a brand new hashtag is simple by putting the hash before a series of words, and if it hasn't been used before, a new hashtag is invented.

1.1.3. Origin of Implementation

The main objective is to implement machine learning based hashtag segmentation application.

The first work was using twitter developer tools to extract tweets from Tweeter's database. In the case of hashtag extraction, there are several issues. Using large amount of raw data that is recieved from social media is the way of creating corpus. In the field of natural language processing, the essential requirements are datasets. Having realiable training and test data helps to improve current algorithm. Because languages are flexible and few training and test data cause to reproduce wrong idea. It somehow clarifies why there is not enough research in Turkish. That is because, improving datasets might help researchers to test their idea and models in the future.

The starting point of project is primarily based upon gathering sufficient data to avoid backing to drawing point. That is, being blind to quantity of data causes to quit idea. Hence, the researcher should collect large amount of data, but also with well-selected contents. When the research topic comes to natural language processing, size and quantity of data is important. Extending corpus enhances current models to achieve better results.

1.1.4. Word Segmentation

Word segmentation means dividing a text into meaningful words. Human-beings can divide text into words with their mental process, but computer not. Word segmentation became more difficult, meanwhile not using a seperator or using more than one form for seperation. Hence, natural language processing begins after that field.

There are several methods about word segmentation. Methods will be discussed in the case of convenience with Turkish.

1.2. Motivation

The common problem about languages which have Latin alphabet system is deciding the word boundary. There are three types of word boundary type.

First one is using space between words. We can not use space segmentation in our project, because hashtags do not contain spaces. Second type is using uppercase letters at the beginning of words or using underscore between the words. It is the main seperation rule of our word segmentation. If hashtags have more than one word in it, we can check the uppercase letters or underscores to decide word boundaries. However; when it comes to real world, the usage of letters in hashtags differs from the second type. The third one is using no uppercase or using nonsense uppercases in hashtags. Because of the natural languages' aspects, datasets contain the hashtags which are the third type of boundary type.

For natural language processing, we have to determine the word's boundaries first. The method that we used tries to work on collected data to create a model that demonstrates the Turkish words' structures.

2. STATE OF ART

2.1. Related Papers and Projects

There are no paper or research about hashtag segmentation in Turkish. So that, the discussion will be based upon other versions of hashtag segmentation and as well as word segmentation.

The problem of word segmentation has been studied in various contexts. One of the most frequently used method for that is maximum matching, by Wong and Chan 1996. Many approaches help us to handle unknown words and to get best proper answer in the case of ambiguity. Venkataraman[1] uses unsupervised word segmentation techniques for finding words in automatically transcribed speech. Recently, Macherey et al.[2] use unsupervised techniques for word segmentation, highlighting the application of this technique to multiple European languages. The work of Wang et al.[3] use unsupervised techniques with multiple corpora for word segmentation.

2.1.1. Exploring the Use of Hashtag Segmentation and Text Quality Ranking

A common practice in tweets is to identify their subject topic by means of a hashtag. In this project, they have given information that a hashtag cannot contain white spaces. They also noticed that people usually concatenate

[1]A. Venkataraman, A statistical model for word discovery in transcribed speech. Computational Linguistics, 2001.

[2]K. Macherey, A. M. Dai, D. Talbot, A. C. Popat, and F. Och. Language-independent compound splitting with morphological operations. In ACL HLT, 2011.

[3]K. Wang, C. Thrasher, and B.-J. P. Hsu. Web scale nlp: a case study on url word breaking. In Proceedings of the 20th international conference on World wide web, New York, NY, USA, 2011. ACM.

more words together, to form a short phrase. They also mentioned that the distinct words composing a hashtag is not a simple task to be automatized.

It is obvious that our project and their project are based on the same aspects like; "Some users identify the distinct words using a CamelCase style, i.e., capitalizing the first letter of each word, other leave the words all in lowercase or uppercase. Other use underscores _ to separate words, but it is not a common case because is wastes characters. Usually words are just juxtaposed without any evident or coherent use of separation signs and actually there are no common rules one can rely on to segment hashtags."[4] They demote their problem as word segmentation problem. The next figure represents the working principles of their system. That system uses the same learning data path as we have.

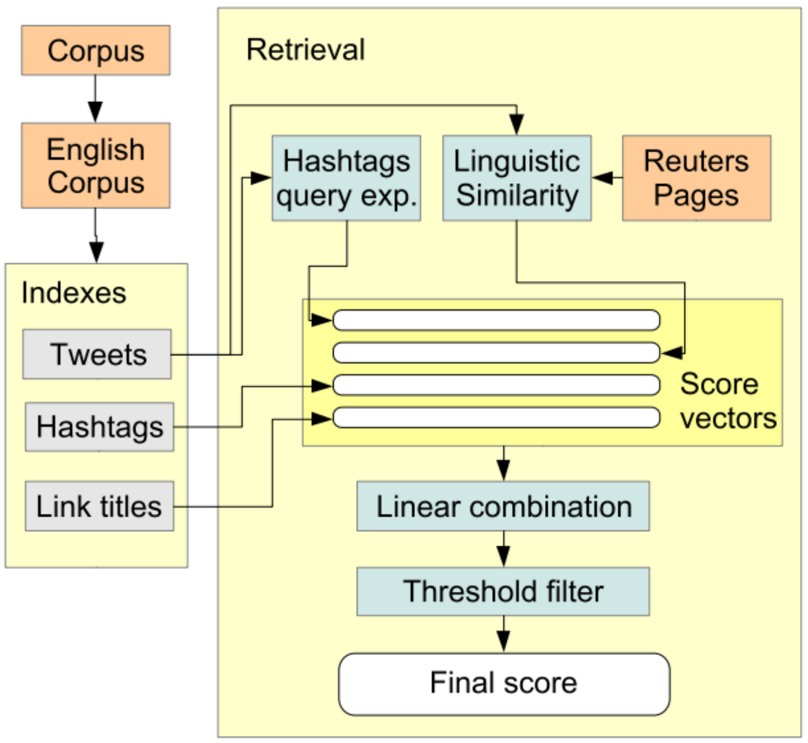


FIGURE 1. Modules of System

They used the words distribution model and a hashtag, the hashtag segmentation module converts the hashtag to a vector of words composing them. Finally, the last figure related to that paper is about how big dataset's size is.

2.1.2. Segmenting Web-Domains and Hashtags using Length Specific Models

In this project, they study two applications in the internet domain. First application is the web domain segmentation which is crucial for monetization of broken URLs. Secondly, they propose and study a novel application of twitter hashtag segmentation for increasing recall on twitter searches. They remark that existing methods for word segmentation use unsupervised language models.

They have realized that when using multiple corpora, the joint probability model from multiple corpora performs significantly better than the individual corpora during the project. Motivated by this, they propose weighted

[4]Giacomo Berardi, et al., ISTI@TREC Microblog track 2011: exploring the use of hashtag segmentation and text quality ranking

	total	effective	retweets	null	hashtags	users
Entire corpus	16.141.812	13.812.346	1.104.780	1.224.686	655.850	5.356.842
English set	4.510.329	4.068.158	442.171		288.753	2.021.759
Hashtags subset	791.464	640.870	150.594		288.753	514.401
Link subset	676.957	674.471	2.486		14.399	400.631

FIGURE 2. Some statistics from the corpus and the subsets that is selected for indexing, the total number of tweets is divided in effective tweets, retweets and null tweets. The hashtags column indicates the number of unique hashtags.

joint probability model, with weights specific to each corpus. Finally, they observed that length of segments is an important parameter for word segmentation. The length specific models further improve segmentation accuracy over supervised probability models.

2.1.3. A Simple and Effective Unsupervised Word Segmentation Approach

In this paper, they propose a new unsupervised algorithm to achieve word segmentation. The main idea of their approach is a novel word induction criterion which is similar with second paper. As they explain; "We devise a method to derive exterior word boundary information from the link structures of adjacent word hypotheses and incorporate interior word boundary information to complete the model."[5] They have also worked with Chinese datasets to claim their approach. If it is working with even difficult language systems, it will be achieved with other systems. They also reported that their approach is simpler and more efficient than the Bayesian methods and more suitable for real-world applications.

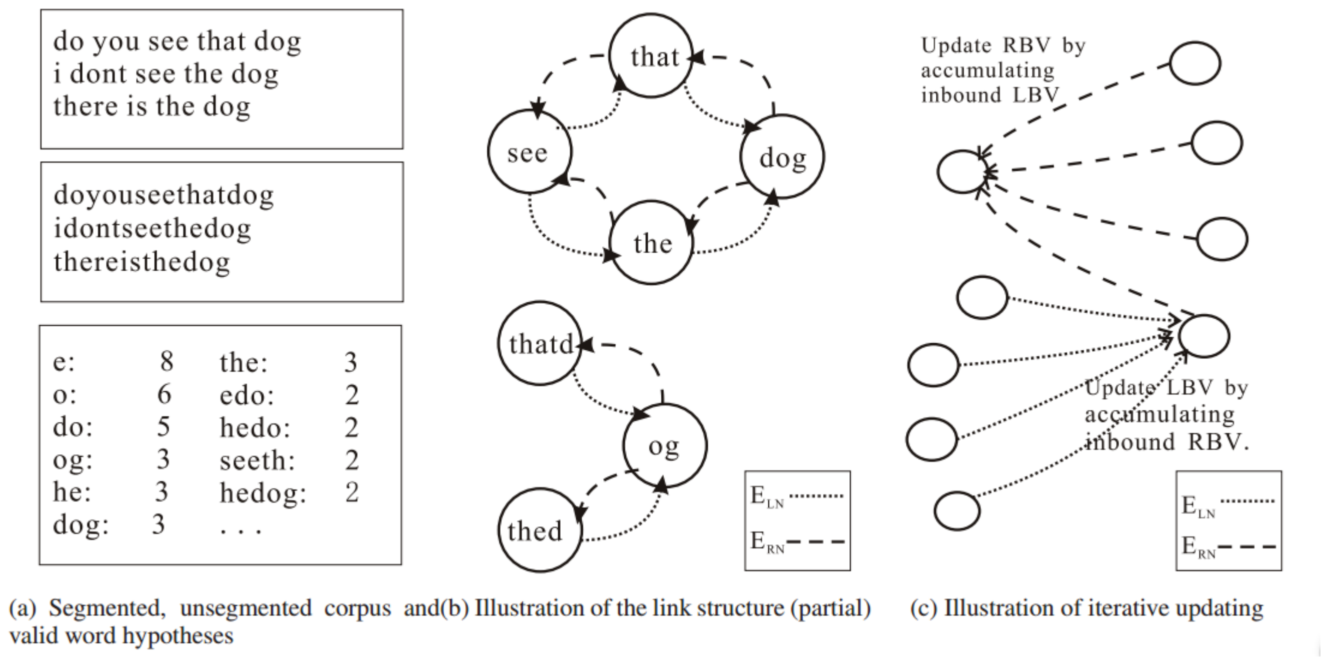

do you see that dog
i dont see the dog
there is the dog
doyouseethatdog
idontseethedog
thereisthedog
e: 8 the: 3
o: 6 edo: 2
do: 5 hedo: 2
og: 3 seeth: 2
he: 3 hedog: 2
dog: 3 . . .
that
see
dog
the
thatd
og
thed
E_LN
E_RN
Update RBV by accumulating inbound LBV
Update LBV by accumulating inbound RBV.
(a) Segmented, unsegmented corpus and valid word hypotheses
(b) Illustration of the link structure (partial)
(c) Illustration of iterative updating

FIGURE 3. Illustrations of constructing the link structures of word hypotheses and calculating the exterior boundary values.

[5]A Simple and Effective Unsupervised Word Segmentation Approach, Songjian Chen, Yabo Xu, Huiyou Chang

2.1.4. URL Segmentation

Another approach is word segmentation of URL links. To think about the word segmentation and recognition problem for URL links, reasearchers adopt some basic principles from rulebased approach.

The dictionary should have sufficient amount of word entries. However, the occurence of compound words makes it very difficult to match every component string exactly with the dictionary entries. Evaluation of the hashtag segmentation has started to be improved with search engine improvements of Twitter.

Finite state transducer is an approach for title token base URL segmentation. It splits and segments according to previous-seen web page title simultaneously. For example, URL link includes ”cs” that might correspond to ”computer science” in many training page's title. If ”cs” is encountered in the testing corpus, it is automatically expanded to ”computer science”. The tranducer has several rules that give score. This score can be obtain by certain moves which match or skip letters in the tokens of title with corresponding letters. Expansion can be valid if it covers all letters in the segment.

2.2. Improvement of Our Project

In our project, there are several improvements according to local and global researches. Firstly, as we mentioned before, it is the first project that introduces a platform for segmentation of Twitter hashtags in Turkish. We also gathered datasets from Twitter and parse them into ”tweet body” and ”hashtags”. We have used maximum entropy model to separate words. We have also imlement an application feature to embed project into any platform. If we get hashtags, we can return the segmented version of them

3. METHODS

3.1. Current Methods

We will discuss two types of models and both models focus on words, not boundaries. And they also use little or no domain-specific information.

3.1.1. Parser Model

No special mechanism is needed for word segmentation; it results from interaction of perception and internal representation. Humans are not tracking boundary statistics; segmentation results from general properties of attention, perception, and memory.

Initially, input is perceived and chunked randomly into units;

- Units are encoded in memory.
- Memory decays rapidly.
- Uncommon units disappear, common units are reinforced.
- Units in memory influence perception and encoding of new input (input is segmented into existing units).

The properties of parser model;

- No explicit tracking of statistics is needed.
- Works on experimental stimuli but might need modifications for realistic language.
- Probably would work in many domains.

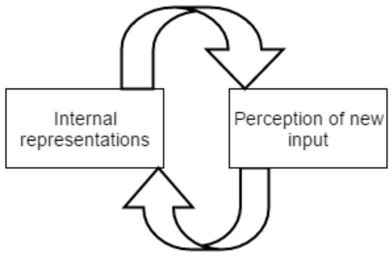


FIGURE 4. Parser System

3.1.2. Bayesian Model

Bayesian model has two varieties in the case of words, that is, whether they are independent from eachother or dependent on other words. If they are independent, we are talking about unigram models. However, if words are dependent on others from same corpus, we are talking about bigram models.

The working data is unsegmented corpus for bayesian models. This hypotheses works on sequences of word tokens. It introduces optimal solution with highest prior probability.

The properties of bayesian model;

- Good segmentations of naturalistic data can be found using fairly weak/domain-general prior assumptions.
- Utterances are composed of discrete words.
- Units tend to be short.
- Some units occur frequently, most do not.
- Units tend to come in predictable patterns.
- More sophisticated use of information works better.

3.2. Our Method

We used "git" as a version control system. Our version history is as follows.

Date	Action
17 October, 2015	Twitter API update is done.
8 November, 2015	Datasets are created and parsing them into database is completed.
9 November, 2015	First Report (Midterm Report) and database upgrades(tweet body and hashtags are seperated) are done.
14 November, 2015	Midterm Report is finished.
15 December, 2015	Hashtag verification is extended.
16 December, 2015	Random hashtags are created for test data.
9 January, 2016	Four features are implemented for learning data(Maximum Entropy Model).
10 January, 2016	Maximum Entropy Model is created with current learning data.
12 January, 2016	Final Project Report, video presentation, application and poster are added.
13 January, 2016	Final delivery is done.

When we started working on the project, there were no datasets for machine learning algorithm and some testing features. Very beginning of the project, we have decided to use "python" as main programming language, because we work on languages, that is, contains a lot of text data. It is knowns off as scripting language. There are several scripting languages as perl, ruby etc. But in the case of size of community and finding efficient APIs, python has more power on them.

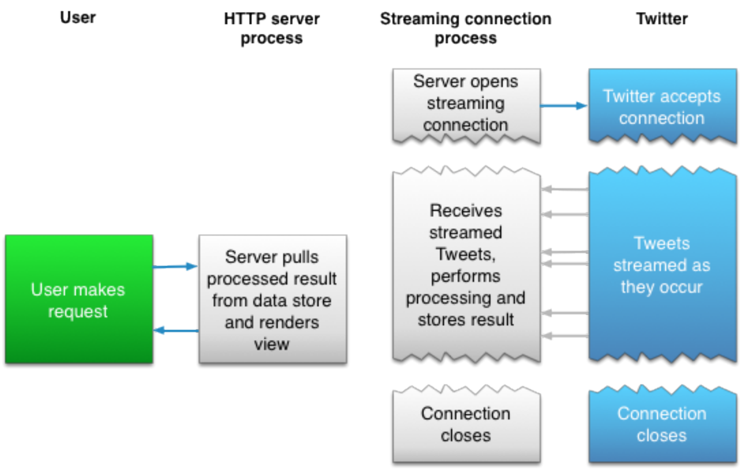
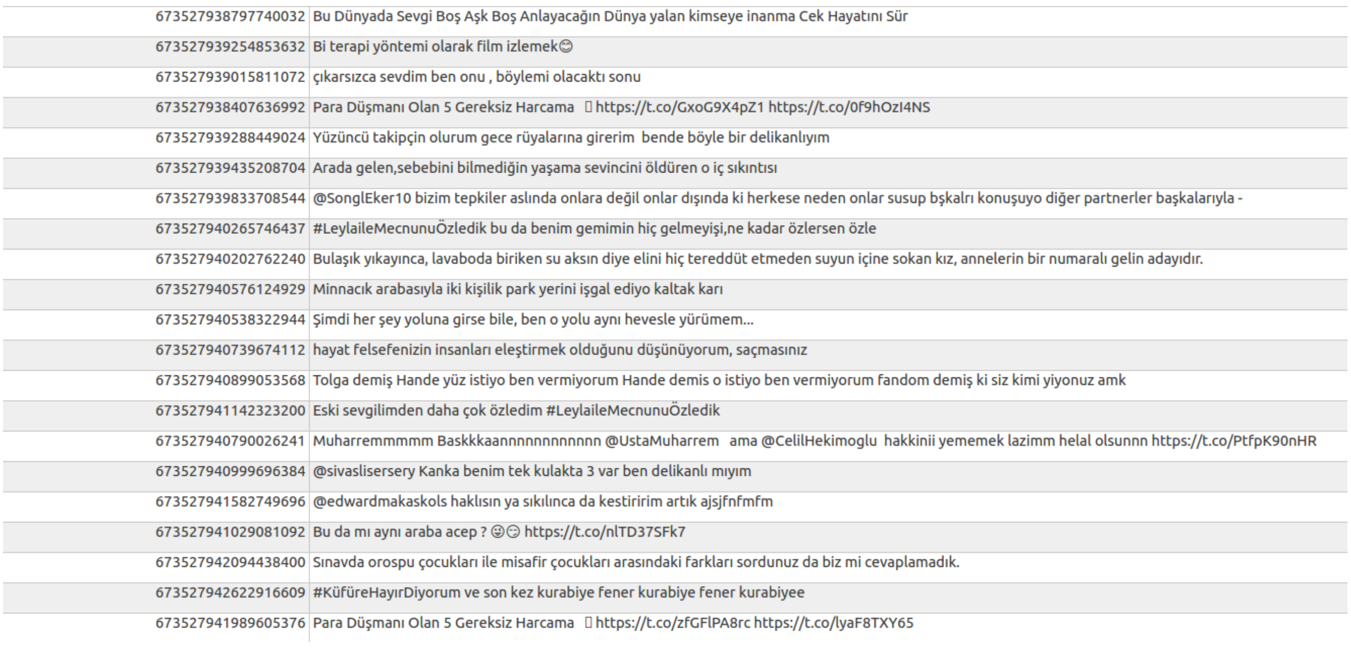

User
HTTP server process
Streaming connection process
Twitter
Server opens streaming connection
Twitter accepts connection
User makes request
Server pulls processed result from data store and renders view
Receives streamed Tweets, performs processing and stores result
Tweets streamed as they occur
Connection closes
Connection closes

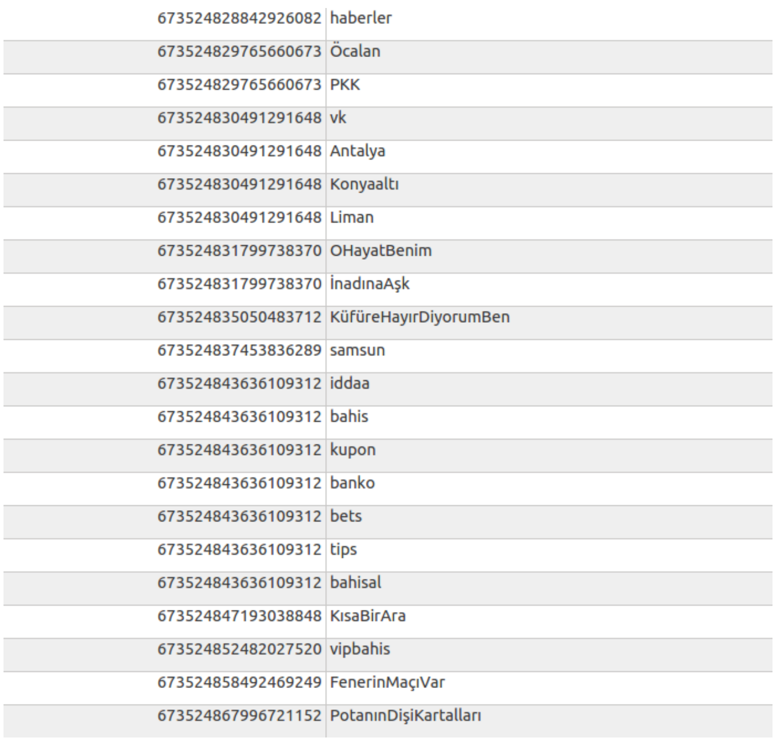
FIGURE 5. Twitter Streaming API

We started to gather data from "Twitter" via their developer accounts. We create a Twitter account and sign up for developer account. We get developer keys and embed them into streaming Twitter API which is "tweepy" After that, we began to collect tweets in ".json" format which includes lots of information about tweets. We decided to extract tweet bodies and ID's.



673527938797740032	Bu Dünyada Sevgi Boş Aşk Boş Anlayacağın Dünya yalan kimseye inanma Cek Hayatını Sür
673527939254853632	Bi terapi yöntemi olarak film izlemek😊
673527939015811072	çıkarsızca sevdim ben onu , böylemi olacaktı sonu
673527938407636992	Para Düşmanı Olan 5 Gereksiz Harcama https://t.co/GxoG9X4pZ1 https://t.co/0f9hOzI4NS
673527939288449024	Yüzüncü takipçin olurum gece rüyalarına girerim bende böyle bir delikanlıyım
673527939435208704	Arada gelen,sebebini bilmediğin yaşama sevincini öldüren o iç sıkıntısı
673527939833708544	@SonglEker10 bizim tepkiler aslında onlara değil onlar dışında ki herkese neden onlar susup bşkalrı konuşuyo diğer partnerler başkalarıyla -
673527940265746437	#LeylaileMecnunuÖzledik bu da benim gemimin hiç gelmeyişi,ne kadar özlersen özle
673527940202762240	Bulaşık yıkayınca, lavaboda biriken su aksın diye elini hiç tereddüt etmeden suyun içine sokan kız, annelerin bir numaralı gelin adayıdır.
673527940576124929	Minnacık arabasıyla iki kişilik park yerini işgal ediyo kaltak karı
673527940538322944	Şimdi her şey yoluna girse bile, ben o yolu aynı hevesle yürümem...
673527940739674112	hayat felsefenizin insanları eleştirmek olduğunu düşünüyorum, saçmasınız
673527940899053568	Tolga demiş Hande yüz istiyo ben vermiyorum Hande demis o istiyo ben vermiyorum fandom demiş ki siz kimi yiyonuz amk
673527941142323200	Eski sevgilimden daha çok özledim #LeylaileMecnunuÖzledik
673527940790026241	Muharremmmmm Baskkkaannnnnnnnnnnn @UstaMuharrem ama @CelilHekimoglu hakkinii yememek lazimm helal olsunnn https://t.co/PtfpK90nHR
673527940999696384	@sivaslisersery Kanka benim tek kulakta 3 var ben delikanlı mıyım
673527941582749696	@edwardmakaskols haklısın ya sıkılınca da kestiririm artık ajsjfnfmfm
673527941029081092	Bu da mı aynı araba acep ? 😃😊 https://t.co/nlTD37SFk7
673527942094438400	Sınavda orospu çocukları ile misafir çocukları arasındaki farkları sordunuz da biz mi cevaplamadık.
673527942622916609	#KüfüreHayırDiyorum ve son kez kurabiye fener kurabiye fener kurabiyee
673527941989605376	Para Düşmanı Olan 5 Gereksiz Harcama https://t.co/zfGFlPA8rc https://t.co/lyaF8TXY65

FIGURE 6. Tweet body examples from training corpus



673524828842926082	haberler
673524829765660673	Öcalan
673524829765660673	PKK
673524830491291648	vk
673524830491291648	Antalya
673524830491291648	Konyaaltı
673524830491291648	Liman
673524831799738370	OHayatBenim
673524831799738370	İnadınaAşk
673524835050483712	KüfüreHayırDiyorumBen
673524837453836289	samsun
673524843636109312	iddaa
673524843636109312	bahis
673524843636109312	kupon
673524843636109312	banko
673524843636109312	bets
673524843636109312	tips
673524843636109312	bahisal
673524847193038848	KısaBirAra
673524852482027520	vipbahis
673524858492469249	FenerinMaçıVar
673524867996721152	PotanınDişiKartalları

FIGURE 7. Tweet hashtag examples from training corpus

After extracting action, we gathered hashtags from bodies in the way of hashtag rules. Meanwhile extracting hashtags from bodies, we stored them (body, ID, hashtags) into database. We used "sqlite" as database tool.

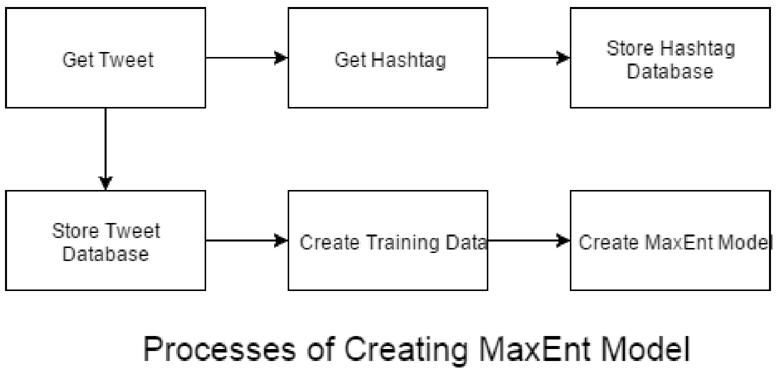


FIGURE 8. The Way of Implementation

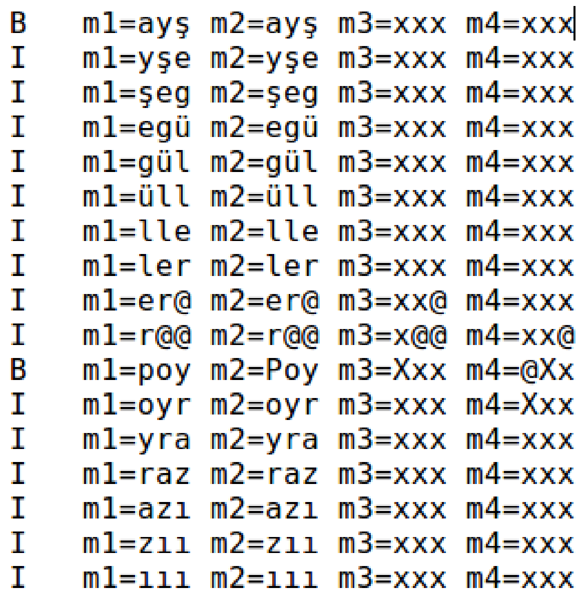
In order to create learning data, we used tweet bodies from our corpus. URLs, hashtags and some parts as well as "@" symbols are removed. We put them together if there is a posibility to have a meaningful parts. In order to mark word boundaries we used " " as our marker. This action helped us to create useful learning data.

After created learning data, we have decided to use four features for our model. Feature mechanism is unique for models. Features can be extended to improve our results. In addition to that, vocabulary of corpus may be used to get better results.

Features are determined in terms of ease of implementation. Getting convenient results can be only improved on working systems.

They are as follows;

- m1: It has current and next two characters as lower case. It includes ”@” in the case of out of bounds.
- m2: It has current and next two characters as no lower or uppercase action. It includes ”@” in the case of out of bounds.
- m3: It checks current and next two characters. It writes ”x” for lowercase, ”X” for uppercase and ”@” for out of bounds situation.
- m4: It checks previous, currend and next characters. It writes ”x” for lowercase, ”X” for uppercase and ”@” for out of bounds situation.



```
B    m1=ayş m2=ayş m3=xxx m4=xxx
I    m1=yşe m2=yşe m3=xxx m4=xxx
I    m1=şeg m2=şeg m3=xxx m4=xxx
I    m1=egü m2=egü m3=xxx m4=xxx
I    m1=gül m2=gül m3=xxx m4=xxx
I    m1=üll m2=üll m3=xxx m4=xxx
I    m1=lle m2=lle m3=xxx m4=xxx
I    m1=ler m2=ler m3=xxx m4=xxx
I    m1=er@ m2=er@ m3=xx@ m4=xxx
I    m1=r@@ m2=r@@ m3=x@@ m4=xx@
B    m1=poy m2=Poy m3=Xxx m4=@Xx
I    m1=oyr m2=oyr m3=xxx m4=Xxx
I    m1=yra m2=yra m3=xxx m4=xxx
I    m1=raz m2=raz m3=xxx m4=xxx
I    m1=azı m2=azı m3=xxx m4=xxx
I    m1=zıı m2=zıı m3=xxx m4=xxx
I    m1=ııı m2=ııı m3=xxx m4=xxx
```

FIGURE 9. Our feature example from training corpus

In order to create test data, we get 1000 random hashtags from our database. We seperate words manually. Finally, we have one original test data and one manually seperated test data to calculate precision, recall, f-measure and accuracy.

3.2.1. Maximum Entropy Model

The target of statistical modeling is to contruct a model that fits best accounts for training data. More spesifically, for given training data, we have probability distrubution. We want to build a model that is close to training probability as possible.

We have used ”Maximum Entropy Model” to decide boundaries of words. The Maximum Entropy model can be introduced as; ”The modeler can choose arbitrary feature functions in order to reflect the characteristic of the problem domain as faithfully as possible. The ability of freely incorporating various problem-specific knowledge in terms of feature functions gives ME models the obvious advantage over other learn paradigms, which often suffer from strong feature independence assumption (such as naive bayes classifier).”[6]

[6]Le, Zhang, Maximum Entropy Modeling Toolkit for Python and C++, 29th December 2004

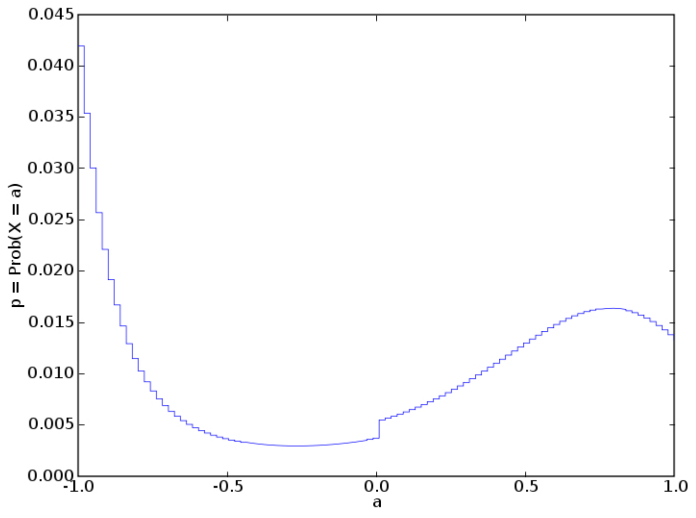


FIGURE 10. Maximum Entropy Model Distribution

We created Model with training data via maximum-entropy model. We create features for original test data. After that, we give that data to maximum-entropy tool and in terms of results from tool, hashtags are segmented. We added segmented hashtags which are calculated by tool and manually segmented hashtags to text file. It helps us to calculate precision, recall, f-measure and accuracy of segmentation method.

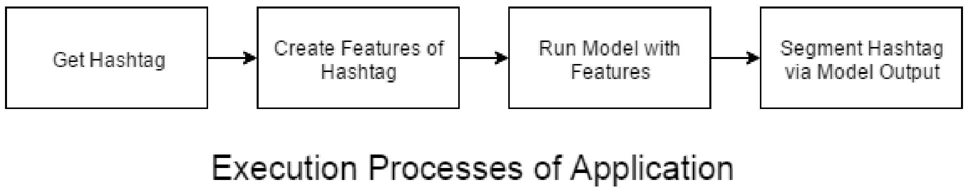

Get Hashtag
Create Features of Hashtag
Run Model with Features
Segment Hashtag via Model Output
Execution Processes of Application

FIGURE 11. Methods of Segmentation

4. RESULTS

In brief, we have collected the tweets from twitter. We tokenize tweets, normalize them and get hashtags from them. We have stored all information related to tweets. That will help us to recognize training and test data. The training and the test data were the part of the current data. We created model from learning data and boundaries of words are decided. At the end of the progression we calculated the measurements of result.

Precision	75.51
Recall	64.80
F-Measure	69.75
Accuracy	49.57
# of Unique Tweets	1.169.282
# of Unique Hashtags	24.003

5. DISCUSSION AND CONCLUSION

5.1. Discussion

There are several methods to achieve word segmentation. As we mentioned, Maximum Entropy Model is used. We may use different methods, but we decided max-ent that gives us best results.

5.2. Conclusion

We have researched our project and get respectable amount of results. Our results are good enough but it can always be improved by computer science developers. It is an open source project and we hope it will help researches in natural language processing field.

6. FUTURE WORK

Future works to increase segmentation results can be as follows;

- Vocabulary of corpus feature can be added.
- More than four features can be added.
- The size of corpus can be extended.
- The irrelevant data can be removed from corpus.
- The tweets which contain foreign words as hashtags can be removed.
- More efficient application can be developed.

7. REFERENCES

REFERENCES

[1] Giacomo Berardi, Andrea Esuli, Diego Marcheggiani, Fabrizio Sebastiani *ISTI@TREC Microblog track 2011: exploring the use of hashtag segmentation and text quality ranking* Istituto di Scienza e Tecnologie dellInformazione Consiglio Nazionale delle Ricerche 56124 Pisa, Italy

[2] Sriram Srinivasan, Rudrasis Chakraborty *Segmenting Web-Domains and Hashtags using Length Specific Models* Indian Statistical Institute

[3] Songjian Chen, Yabo Xu, Huiyou Chang *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence, A Simple and Effective Unsupervised Word Segmentation Approach* School of Information Science and Technology Sun Yat-sen University Guangzhou, 510006, China

[4] P. Analytics *Twitter study, 2009* Retrieved 03 31, 2010, from http://www.scribd.com/doc/18548460/Pear-Analytics-Twitter-Study-August-2009.

[5] Berardi, Giacomo; Esuli, Andrea; Marcheggiani, Diego and Sebastiani, Fabrizio. *Exploring the use of hashtag segmentation and text quality ranking Istituto di Scienza e Tecnologie dellInformazione Consiglio Nazionale delle Ricerche* 56124 Pisa, Italy

[6] Bansal, P., Bansal, R. and Varma V. 2015. *Towards Deep Semantic Analysis Of Hashtags.*

[7] Xue, N. 2003. *Chinese word segmentation as character tagging. International Journal of Computational Linguistics and Chinese Language Processing volume 8*

[8] Rebecca Hiscott. 2013. *The Beginner's Guide to the Hashtag*

8. APPENDIX

```
#Import the necessary methods from tweepy library
from tweepy.streaming import StreamListener
from tweepy import OAuthHandler
from tweepy import Stream

#Variables that contains the user credentials to access Twitter API
access_token = "***********************"
access_token_secret = "****************"
consumer_key = "***********************"
consumer_secret = "********************"


#This is a basic listener that just prints received tweets to stdout.
class StdOutListener(StreamListener):

    def on_data(self, data):
        print (data)
        return True

    def on_error(self, status):
        print (status)


if __name__ == '__main__':

    #This handles Twitter authetification and the connection to Twitter Streaming API
    l = StdOutListener()
    auth = OAuthHandler(consumer_key, consumer_secret)
    auth.set_access_token(access_token, access_token_secret)
    stream = Stream(auth, l)

    stream.filter(track = ["bir","bu","ne","ve","ben","de","evet","var","ama","da",
        "sen","daha","bana","kadar","seni","beni","iyi","tamam","onu","bunu","gibi",
        "yok","benim","her","sana","ki","sadece","neden","burada","senin","ya","zaman",
        "sonra","en","mu","misin","hadi","biraz","musun","ona","bak","oldu","hey","istiyorum",
        "geri","onun","bile","kim","bay","yani","bilmiyorum","buraya","belki","peki","olarak",
        "tek","efendim","biri","haydi","olur","et","olacak","olan","adam","merhaba","orada",
       "herhalde","biz","demek","bilmiyorum","gece","ederim","olmak","o","yapmak","almak",
       "kendi","gelmek","ile","vermek","sonra","insan","iki","el","istemek","ara","son","ilk",
       "veya","bunlar","ancak","yemek","su","girmek","orta","yan","durum","gerekmek","bulunmak",
       "ise","diye","ev"], languages=["tr"])
   #stream.filter(locations=[25.40,44.48,35.51,42.06])

#!/usr/bin/python
# -*- coding: utf8 -*-
import sqlite3
import json

print "What␣do␣you␣want␣to␣do?"
print "[1]␣INSERT␣DATASETS␣TO␣DATABASE"
print "[2]␣CREATE␣DATABASE"

option = input("␣~␣")

conn = sqlite3.connect('tweets.db')

def insertToTexts(docId, body):

       c = conn.cursor()
       c.execute('insert␣or␣ignore␣into␣TEXTS␣values␣(?,?)', (docId, body))
       conn.commit()

def insertToHashtags(docId, hasht):

       c = conn.cursor()
       c.execute('insert␣or␣ignore␣into␣HASHTAGS␣values␣(?,?)', (docId, hasht))
       conn.commit()

if option == 2:

       conn.execute('''CREATE␣TABLE␣TEXTS␣(ID␣INT␣PRIMARY␣KEY␣NOT␣NULL,␣BODY␣CHAR(200));''')

       conn.execute('''CREATE␣TABLE␣HASHTAGS␣(ID␣INT␣NOT␣NULL,␣HASHTAG␣CHAR(100)␣UNIQUE);''')

       print "Table␣created␣successfully";
```

```
    conn.close()

if option == 1:

    print "Which␣dataset␣do␣you␣want␣to␣insert?"

    setNum = raw_input("␣~␣")
    s = ’dataset’ + setNum + ’.json’
    tweets_data_path = s

    tweets_data = []
    tweets_file = open(tweets_data_path, "r")

    for line in tweets_file:
        try:
            tweet = json.loads(line)
            tweets_data.append(tweet)
        except:
            continue

    for tw in tweets_data:
        if not ’limit’ in tw:
            tw_text = tw[’text’]
            if not tw_text.startswith(’RT’):
                tw_text = tw_text.replace(’\n’, ’␣’)
                tw_text = tw_text + ’␣’
                tw_id = tw[’id’]

                insertToTexts(tw_id,tw_text)
                space_index = [pos for pos, char in enumerate(tw_text) if char == ’␣’]

                counter = 0
                for k in range(0,len(space_index)):

                    string = tw_text[counter:space_index[k]]
                    if string.startswith(’#’):
                        string = string[1:]
```

```
                                                if len(string) > 0 and not ’#’ in string:
                                                        if string.startswith(’_’) and len(string) == 1:
                                                                counter = space_index[k] + 1
                                                        else:
                                                                index = 0
                                                                for s in string:
                                                                        if not s in u"_abcdefghijklmnoprstuvyzAB
                                                                                break
                                                                        else:
                                                                                index = index + 1
                                                                hashtag = string[0:index]
                                                                if len(hashtag) > 0:
                                                                        insertToHashtags(tw_id,hashtag)
                                                                counter = space_index[k] + 1
                                                else:
                                                        counter = space_index[k] + 1
                                        else:
                                                counter = space_index[k] + 1
conn.close()
```

```
#!/usr/bin/python
# -*- coding: utf8 -*-
import sqlite3
import codecs

f = codecs.open(’randomHastag.txt’, ’w’, ’utf-8’)
conn = sqlite3.connect(’tweets.db’)
c = conn.cursor()
c.execute("SELECT␣*␣FROM␣HASHTAGS")
rows = c.fetchall()
counter = 0
for row in rows:
        if len(row[1]) > 16:
                f.write(row[1])
                f.write(’\n’)
                counter = counter + 1
#print counter
f.close()
conn.close()
```

```
import json
import sqlite3
import string
from subprocess import call

f = open('training_maxent.txt', 'w')
conn = sqlite3.connect('tweets.db')
c = conn.cursor()
c.execute("SELECT␣*␣FROM␣TEXTS")
rows = c.fetchall()
tweetBodies = []
ourChars = string.punctuation +
"abcdefghijklmnoprstuvyzABCDEFGHIJKLMNOPRSTUVYZ0123456789"

def prepareFeatures(text):

        textSize = len(text)
        for i in range(0, textSize):
                if text[i] != "~":
                        if text[i-1] == "~":
                                f.write("B\t")
                        else:
                                f.write("I\t")
                        # Feature 1
                        f.write("m1=")
                        for j in range(0,3):
                                if i+j < textSize:
                                        if text[i+j] != "~":
                                                if text[i+j] in "0123456789":
                                                                f.write(text[i+j])
                                                else:
                                                        f.write(text[i+j].lower())
                                        else:
                                                if i+j+1 < textSize:
                                                        if text[i+j+1] in "0123456789":
                                                                f.write(text[i+j+1])
                                                        else:
                                                                f.write(text[i+j+1].lower())
```

```
            else:
                f.write("@")
    else:
        f.write("@")
f.write("␣")
# Feature 2
f.write("m2=")
for j in range(0,3):
    if i+j < textSize:
        if text[i+j] != "~":
            f.write(text[i+j])
        else:
            if i+j+1 < textSize:
                f.write(text[i+j+1])
            else:
                f.write("@")
    else:
        f.write("@")
f.write("␣")
# Feature 3
f.write("m3=")
for j in range(0,3):
    if i+j < textSize and text[i+j] != "~":
        if text[i+j] in "0123456789":
            f.write("#")
        else:
            if text[i+j].islower():
                f.write("x")
            elif text[i+j].isupper():
                f.write("X")
    elif i+j < textSize and text[i+j] == "~":
        if i+j+1 < textSize:
            if text[i+j+1] in "0123456789":
                f.write("#")
            else:
                if text[i+j+1].islower():
                    f.write("x")
                elif text[i+j+1].isupper():
                    f.write("X")
        else:
            f.write("@")
    else:
        f.write("@")
f.write("␣")
# Feature 4
f.write("m4=")
if i-1 == 0:
    f.write("@")
else:
    if text[i-1] == "~":
        if text[i-2] in "0123456789":
            f.write("#")
        else:
            if text[i-2].islower():
                f.write("x")
            elif text[i-2].isupper():
                f.write("X")
    else:
        if text[i-1] in "0123456789":
            f.write("#")
        else:
            if text[i-1].islower():
                f.write("x")
            elif text[i-1].isupper():
                f.write("X")

if text[i] in "0123456789":
    f.write("#")
else:
    if text[i].islower():
        f.write("x")
    elif text[i].isupper():
        f.write("X")
if i+1 < textSize:
    if text[i+1] == "~":
        if i+2 < textSize:
```

```
                        if text[i+2] in "0123456789":
                            f.write("#")
                        else:
                            if text[i+2].islower():
                                f.write("x")
                            elif text[i+2].isupper():
                                f.write("X")
                    else:
                        f.write("@")
                else:
                    if text[i+1] in "0123456789":
                        f.write("#")
                    else:
                        if text[i+1].islower():
                            f.write("x")
                        elif text[i+1].isupper():
                            f.write("X")
            else:
                f.write("@")
            f.write("\n")



for row in rows:
    tweet = row[1].split()
    for i in range(0, len(tweet)):
        for letter in tweet[i]:
            if letter not in ourChars:
                tweet[i] = tweet[i].replace(letter, "~")
        if "http" in tweet[i]:
            tweet[i] = "~"
        elif "#" in tweet[i]:
            tweet[i] = "~"
        elif "@" in tweet[i]:
            tweet[i] = "~"
        elif "www" in tweet[i]:
            tweet[i] = "~"
        elif ".com" in tweet[i]:
            tweet[i] = "~"
            # Changing one letter in words.
            for item in string.punctuation:
                  tweet[i] = tweet[i].replace(item, "~")

      text = ""
      for i in range(0, len(tweet)):
            if tweet[i] == "~" and len(text) == 0:
                  continue
            elif tweet[i] == "~" and len(text) > 0 :
                  prepareFeatures(text)
                  text = ""
            elif "~" in tweet[i]:
                  text = text + "~" + tweet[i][:tweet[i].find("~")]
                  if len(text) > 1:
                        prepareFeatures(text)
                  text = ""
            elif "~" not in tweet[i]:
                  if len(tweet[i]) > 0:
                        text = text + "~" + tweet[i]
      if len(text) > 0:
            prepareFeatures(text)

f.close()
conn.close()

# Creating Model
call(["maxent", "training_maxent.txt", "-m", "model", "-i", "30", "--gis"])



from subprocess import call


print("What␣do␣you␣want␣to␣do?")


learningData = "deneme.txt"


call(["maxent", learningData, "-m", "model", "-i", "30", "--gis"])


test = "test.txt"
output = "output.txt"
```

```
call(["maxent", "-p", test, "-m", "model", "-o", output])



from subprocess import call

f = open('testMaxent.txt', 'w')
g = open('testHashtag.txt', 'r')

def prepareFeaturesTest(text):

       textSize = len(text)
       for i in range(0, textSize):
              # Feature 1
              f.write("m1=")
              for j in range(0,3):
                     if i+j < textSize:
                            if text[i+j] in "0123456789":
                                          f.write(text[i+j])
                            else:
                                   f.write(text[i+j].lower())
                     else:
                            f.write("@")
              f.write("␣")
              # Feature 2
              f.write("m2=")
              for j in range(0,3):
                     if i+j < textSize:
                            f.write(text[i+j])
                     else:
                            f.write("@")
              f.write("␣")
              # Feature 3
              f.write("m3=")
              for j in range(0,3):
                     if i+j < textSize:
                            if text[i+j] in "0123456789":
                                   f.write("#")
                            else:
```

```
            if text[i+j].islower():
                f.write("x")
            elif text[i+j].isupper():
                f.write("X")
    else:
        f.write("@")
f.write("␣")
# Feature 4
f.write("m4=")
if i-1 < 0:
    f.write("@")
else:
    if text[i-1] in "0123456789":
        f.write("#")
    else:
        if text[i-1].islower():
            f.write("x")
        elif text[i-1].isupper():
            f.write("X")

if text[i] in "0123456789":
    f.write("#")
else:
    if text[i].islower():
        f.write("x")
    elif text[i].isupper():
        f.write("X")
if i+1 < textSize:
    if text[i+1] in "0123456789":
        f.write("#")
    else:
        if text[i+1].islower():
            f.write("x")
        elif text[i+1].isupper():
            f.write("X")
else:
    f.write("@")
f.write("\n")
```

```
lines = g.readlines()

for line in lines:
        line = line.replace("\n", "")
        prepareFeaturesTest(line)

f.close()

call(["maxent", "-p", "testMaxent.txt", "-m", "../maxEntModel/model", "-o", "BIresults.txt"])


testHashtag = open("testHashtag.txt", 'r')

BIresults = open("BIresults.txt", 'r')

segmented = open("manualSegmented.txt", 'r')

output = open("output.txt", 'w')

hastagLines = testHashtag.readlines()
BIlines = BIresults.readlines()
segmentLines = segmented.readlines()

index = 0
indexSeg = 0
for line in hastagLines:
        line = line.replace("\n", "")
        string = line
        space = 0
        for i in range(0, len(line)):
                if "B" in BIlines[i+index] and i != 0:
                        string = string[:i+space] + "␣" + string[i+space:]
                        space += 1
        index += len(line)
        lineSeg = segmentLines[indexSeg].replace("\n", "")
        output.write(lineSeg)
        output.write("\t")
        output.write(string)
        output.write("\n")
        indexSeg += 1
```