CSIT 6910

Independent Project

Making Sense of Images on Twitter

Student: Mingxin CHENG

Student ID: 20387442

Student Email: mchengaa@connect.ust.hk

Supervisor: Prof. Wilfred Siu-Hung NG

Date: May 27th, 2017

Contents

1 Introduction

1.1 Background:

In this era of Mobile Internet and Big Data, billions of people are using Twitter, one of the world's largest SNS, to post news and express themselves. One highlight of Twitter is that users can easily upload images along with their text. It is very interesting to look into the images and tweets on Twitter.
Meanwhile, thanks to the development of big data, database and natural language processing, more and more researchers are doing studying on Twitter.

1.2 Challenge:

In most of the time, the images on Twitter have high correlation with the tweet text. However, not all images are always associated with the relevant text. And making sense of the semantic meaning of an image has always been a challenging task.

1.3 Target:

Thus, what we want to do is to map images and texts on Twitter into the same feature space, making keyword search for untagged images possible. We are going to first solve the problem how to tag the images with meaningful keywords. And then build a keyword search system in which we can search for the newest images from Twitter by giving the keywords.

2 Project Overview

2.1 Overview

The overview of our project *Making Sense of Image on Twitter* is shown as follows:
Our Goal is to build a Keyword Search System for Twitter images.
The first step is to get data from Twitter. We have two ways to fetch data from Twitter. One way is using Twitter streaming api to fetch the data. The other way is that we crawl tweets from Twitter using the tweet crawler we implemented by ourselves. The data we get are mainly in two topics, daily life and news. We choose the tweets of news accounts to build our system.

After getting the data, we need to preprocess the data and extract the useful information we need. We first extract the tweet's url, tweet image's url and the tweet text. After that, we need to extract the keywords from the tweet text using natural language processing.

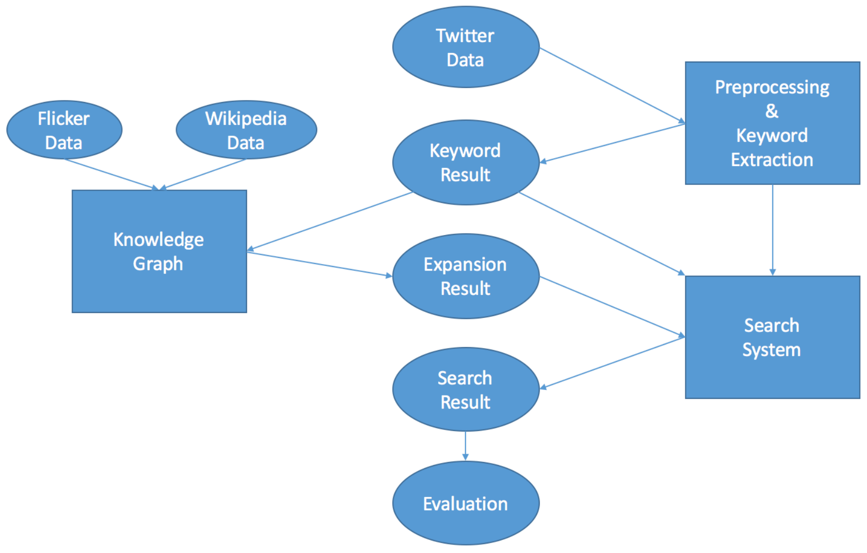


Chart 2-1 Project Overview

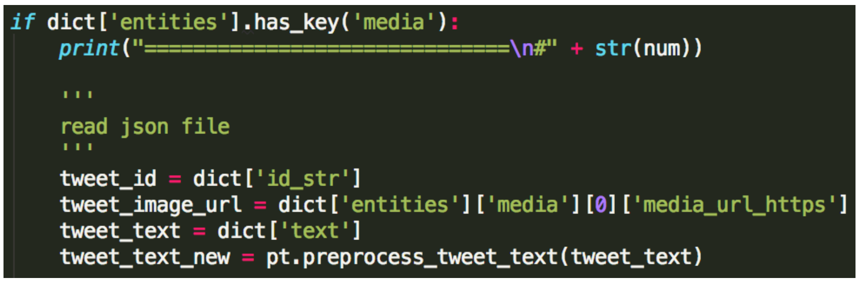
After that, we want to make sense of the tweet. For example, if we have two keywords 'Donald Trump' and 'Hillary Clinton', we want that we can get this tweet by search 'US Election'. So we need to do the keyword expansion. In our project, we expand the keywords using knowledge graph.
Then, we can build the Search System using the Tweet basic information, keywords, expanded keywords.
In the end, we will use some methods evaluate our Search System.

2.2 Work Assigned

In our group's project, I am assigned mainly to do three parts of job.
The first part is to preprocess the Twitter data. I have to get the useful information from the raw Twitter data.
The second part is to do the keyword extraction after getting the preprocessed data. I need to do some research first to learn how to extract the keywords from short text like twitter. Then I have to implement the keyword extraction algorithm by myself. After me finishing my job, my teammates can expand the keywords and build the search system.
The last part of my job is to help to build the evaluation system for our search system.

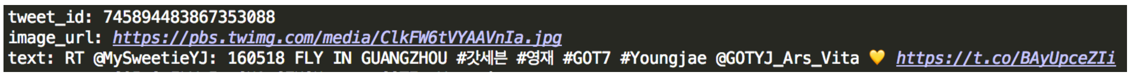
3 Twitter Data Preprocessing

The first part of my job is to preprocess the Twitter data.
The Twitter data we get is encoded in json format. Json is short for JavaScript Object Notation and is a file format that transmits data objects consisting of attribute–value pairs and array data types. So we extract the information we want by the keys. The information we want are:
[tweet_id, tweet_image_url, tweet_text]



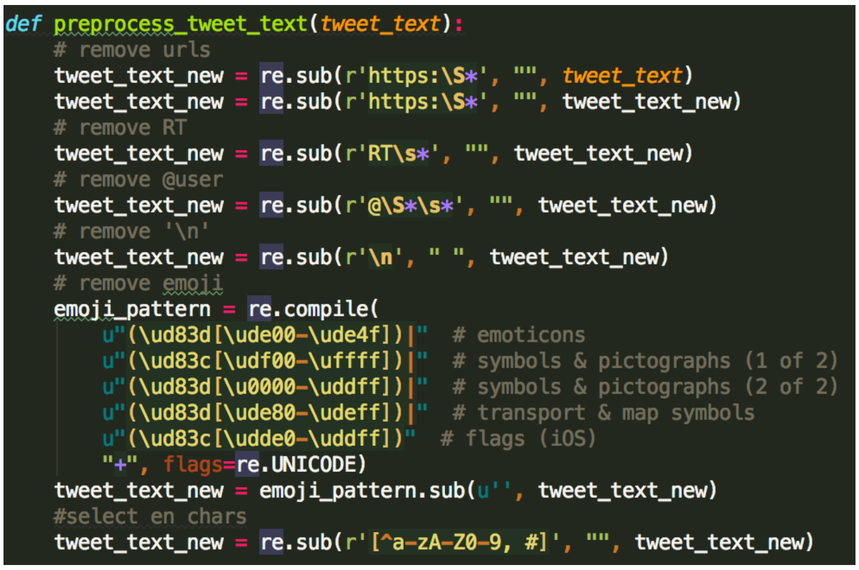
```python
if dict['entities'].has_key('media'):
    print("==============================\n#" + str(num))

    '''
    read json file
    '''
    tweet_id = dict['id_str']
    tweet_image_url = dict['entities']['media'][0]['media_url_https']
    tweet_text = dict['text']
    tweet_text_new = pt.preprocess_tweet_text(tweet_text)
```

After getting the tweet text, we cannot use it directly. For example, in a tweet like this:



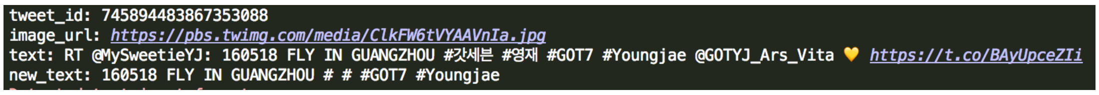
```
tweet_id: 745894483867353088
image_url: https://pbs.twimg.com/media/ClkFW6tVYAAVnIa.jpg
text: RT @MySweetieYJ: 160518 FLY IN GUANGZHOU #갓세븐 #영재 #GOT7 #Youngjae @GOTYJ_Ars_Vita 💛 https://t.co/BAyUpceZIi
```

There are too much noise and useless information in it. We cannot use the raw tweet, so we need to preprocess it.
We decide to use regular expression to remove some useless information like urls, retweet info and at-mentions. Also, we want to remove illegal inputs like ‘\n’ ‘\t’ emojis and other illegal chars.



```python
def preprocess_tweet_text(tweet_text):
    # remove urls
    tweet_text_new = re.sub(r'https:\S*', "", tweet_text)
    tweet_text_new = re.sub(r'https:\S*', "", tweet_text_new)
    # remove RT
    tweet_text_new = re.sub(r'RT\s*', "", tweet_text_new)
    # remove @user
    tweet_text_new = re.sub(r'@\S*\s*', "", tweet_text_new)
    # remove '\n'
    tweet_text_new = re.sub(r'\n', " ", tweet_text_new)
    # remove emoji
    emoji_pattern = re.compile(
        u"(\ud83d[\ude00-\ude4f])|"  # emoticons
        u"(\ud83c[\udf00-\uffff])|"  # symbols & pictographs (1 of 2)
        u"(\ud83d[\u0000-\uddff])|"  # symbols & pictographs (2 of 2)
        u"(\ud83d[\ude80-\udeff])|"  # transport & map symbols
        u"(\ud83c[\udde0-\uddff])"  # flags (iOS)
        "+", flags=re.UNICODE)
    tweet_text_new = emoji_pattern.sub(u'', tweet_text_new)
    #select en chars
    tweet_text_new = re.sub(r'[^a-zA-Z0-9, #]', "", tweet_text_new)
```

After the preprocess ,the new text is like:



```
tweet_id: 745894483867353088
image_url: https://pbs.twimg.com/media/ClkFW6tVYAAVnIa.jpg
text: RT @MySweetieYJ: 160518 FLY IN GUANGZHOU #갓세븐 #영재 #GOT7 #Youngjae @GOTYJ_Ars_Vita 💛 https://t.co/BAyUpceZIi
new_text: 160518 FLY IN GUANGZHOU # # #GOT7 #Youngjae
```

4 Tweet Keyword Extraction

4.1 Related Works

Keyword Extraction has always been a difficult topic in natural language processing. There are two methods for keyword extraction.

4.1.1 Predefine a Vocabulary Library for keywords

The first method is identifying predefined vocabularies that match a given text. In this method, we often predefine the vocabulary library for the certain topic we concentrate on.
For example, if we concentrate on news topic, we can predefine a vocabulary library: {Barack Obama, Donald Trump, Hillary Clinton, US Election…}. Or if we focus on the food topic, we can predefine a vocabulary library: {Noodle, Rice, Bread, …}.
The advantage of this method is this method is easy to implemented. While the disadvantage is that we need to update vocabulary library by adding new words frequently.

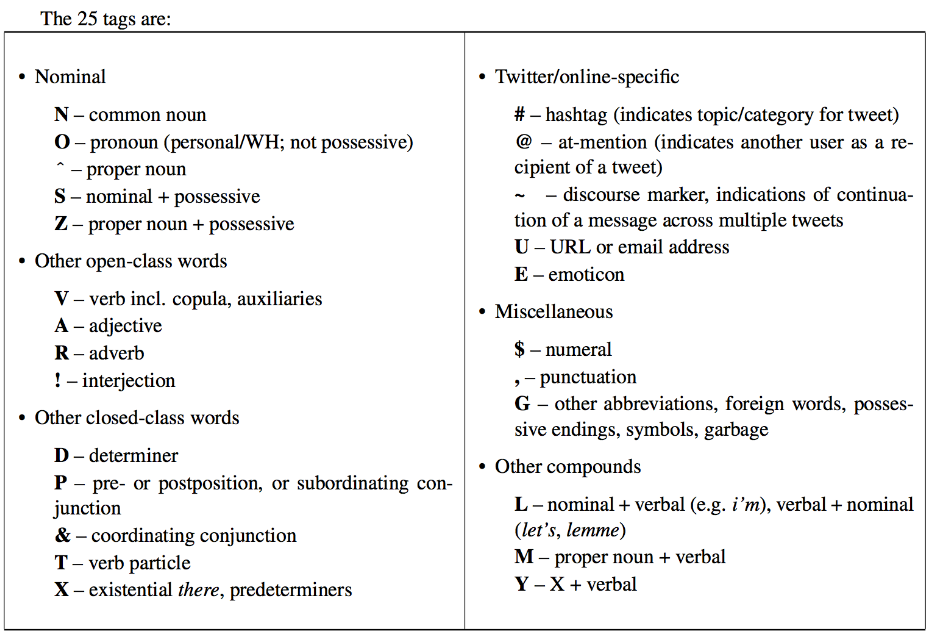
4.1.2 Extract significant words as keywords

The other method is extracting the most significant words and phrases that appear in given text. In this method, algorithms differ in the length of the text.
For long-text algorithm, there are three main components:
Candidate selection: First we extract all possible words, phrases, terms or concepts that can potentially be keywords.
Properties calculation: For each candidate, we calculate properties that indicate that it may be a keyword. For example, a candidate appearing in the title of a book is a likely keyword.
Scoring and selecting keywords: All candidates can be scored by either combining the properties into a formula, or using a machine learning technique to determine probability of a candidate being a keyword. A score or probability threshold, or a limit on the number of keywords is then used to select the final set of keywords.
The common algorithms for long-text keyword extraction are: RAKE, Maui, KEA, etc.
For short-text, we often use parsing method or semantic understanding to spot the keywords.
The common procedure for keyword extraction in short-text is: tokenizing & tagging; Using CFG for phrases extracting and re-tagging; keyword spotting.

4.2 Implementation Details

4.2.1 Tokenizing & Tagging:

The first step of my implementation of keyword extraction is to tokenize and tag the tweet. I use CMU Twitter NLP tool to do the tokenizing and tagging job for its high accuracy for tweets.



The 25 tags are:

- Nominal

  N – common noun
  O – pronoun (personal/WH; not possessive)
  ^ – proper noun
  S – nominal + possessive
  Z – proper noun + possessive

- Other open-class words

  V – verb incl. copula, auxiliaries
  A – adjective
  R – adverb
  ! – interjection

- Other closed-class words

  D – determiner
  P – pre- or postposition, or subordinating conjunction
  & – coordinating conjunction
  T – verb particle
  X – existential *there*, predeterminers

- Twitter/online-specific

  # – hashtag (indicates topic/category for tweet)
  @ – at-mention (indicates another user as a recipient of a tweet)
  ~ – discourse marker, indications of continuation of a message across multiple tweets
  U – URL or email address
  E – emoticon

- Miscellaneous

  $ – numeral
  , – punctuation
  G – other abbreviations, foreign words, possessive endings, symbols, garbage

- Other compounds

  L – nominal + verbal (e.g. *i'm*), verbal + nominal (*let's*, *lemme*)
  M – proper noun + verbal
  Y – X + verbal

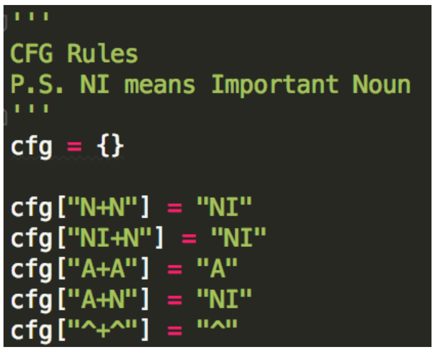
Chart 4-1 CMU Twitter NLP Tokenizing & Tagging Tool

As is shown above, the CMU Twitter NLP Tool is specially designed for tweets. Despite the common seen tags N (common noun), ^ (proper noun), A (adjective), it can tag the hashtags, atmentions, URLs, emorticons with high accuracy.

4.2.2 Context-Free Grammar (CFG)

In formal language theory, a context-free grammar (CFG) is a certain type of formal grammar: a set of production rules that describe all possible strings in a given formal language. Production rules are simple replacements.
For example, the rule
A -> B
Means A can be replaced by B.
Also, there can be multiple replacement rules for any given value. For example,
A + B -> C
Means that A+B can be replaced by C.
In Natural Language Processing, the CFG can be used to extract some certain phrase and tags we want.

In our Twitter Keywords Extraction Project, we arbitrary set the rules:

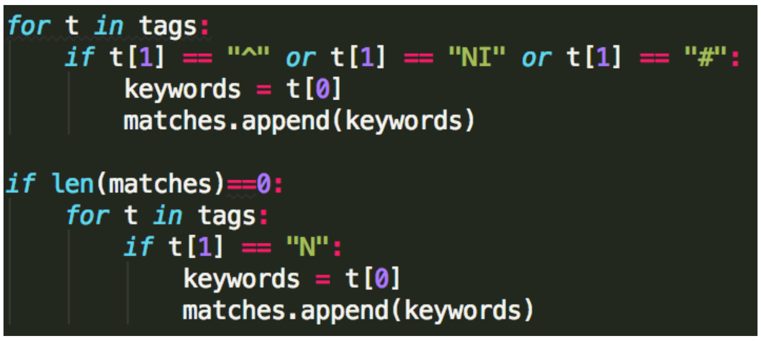


```
'''
CFG Rules
P.S. NI means Important Noun
'''
cfg = {}

cfg["N+N"] = "NI"
cfg["NI+N"] = "NI"
cfg["A+A"] = "A"
cfg["A+N"] = "NI"
cfg["^+^"] = "^"
```

(P.S. NI is a tag we arbitrarily set, meaning important noun phrase)
These rules mean:
A common noun plus a common noun can be seen as an important noun phrase.
An important noun phrase plus a common noun can be seen as an important noun phrase.
An adjective plus an adjective can be seen as a adjective phrase.
An adjective plus a common noun can be seen as an important noun phrase.
A proper noun plus a proper noun can be seen as a proper noun.
Having these rules., we can run the CFG to re-tag the words.

4.2.3 Keyword Spotting

After getting the re-tagged words, we can spot the keywords by some rules.



```
for t in tags:
    if t[1] == "^" or t[1] == "NI" or t[1] == "#":
        keywords = t[0]
        matches.append(keywords)

if len(matches)==0:
    for t in tags:
        if t[1] == "N":
            keywords = t[0]
            matches.append(keywords)
```

We set two rounds to get the keywords.
In the first round, we get all the important noun phrases (NI, generated by adjective plus common noun or common noun plus common noun or adjective plus adjective plus common noun, etc.), proper nouns (^) and hashtags (#).
If we do not get any answers in the first round, we will run the second round.
In the second round, we will extract the Subject and object in the sentence.

4.2.4 Result

I will give two examples to illustrate my implementation of keywords extraction on Twitter.
Sample 1:
Step 1: Tokenizing & Tagging using CMU Twitter NLP Tool:

A	Little	Syrian	girl	survived	a	Russian	napalm	attack
D	A	A	N	V	D	A	N	N
but	will	be	marked	for	life	.	#Assad	#Syria
&	V	V	V	P	N	,	#	#

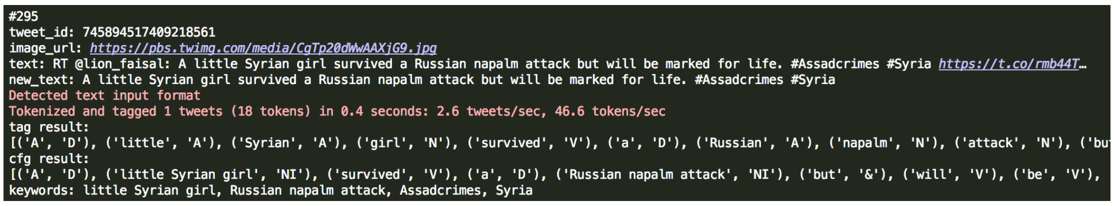
Step 2: Re-tagging using CFG:

A	Little	Syrian	girl	survived	a	Russian	napalm	attack
D	NI			V	D	NI		
but	will	be	marked	for	life	.	#Assad	#Syria
&	V	V	V	P	N	,	#	#

Step 3: Spotting the Keywords:

A	Little	Syrian	girl	survived	a	Russian	napalm	attack
D	NI			V	D	NI		
but	will	be	marked	for	life	.	#Assad	#Syria
&	V	V	V	P	N	,	#	#

Thus, the keywords are:
{Little Syrian girl, Russian napalm attack, Assadcrimes, Syria}
The Console Outputs:



```
#295
tweet_id: 745894517409218561
image_url: https://pbs.twimg.com/media/CgTp20dWwAAXjG9.jpg
text: RT @lion_faisal: A little Syrian girl survived a Russian napalm attack but will be marked for life. #Assadcrimes #Syria https://t.co/rmb44T…
new_text: A little Syrian girl survived a Russian napalm attack but will be marked for life. #Assadcrimes #Syria
Detected text input format
Tokenized and tagged 1 tweets (18 tokens) in 0.4 seconds: 2.6 tweets/sec, 46.6 tokens/sec
tag result:
[('A', 'D'), ('little', 'A'), ('Syrian', 'A'), ('girl', 'N'), ('survived', 'V'), ('a', 'D'), ('Russian', 'A'), ('napalm', 'N'), ('attack', 'N'), ('bu
cfg result:
[('A', 'D'), ('little Syrian girl', 'NI'), ('survived', 'V'), ('a', 'D'), ('Russian napalm attack', 'NI'), ('but', '&'), ('will', 'V'), ('be', 'V'),
keywords: little Syrian girl, Russian napalm attack, Assadcrimes, Syria
```

Sample 2:
Step 1: Tokenizing & Tagging using CMU Twitter NLP Tool:

This	Morning’s	front	page	.	What	will	Tomorrow’s
D	S	N	N	,	O	V	S
bring	?	#EURef					
V	,	#					

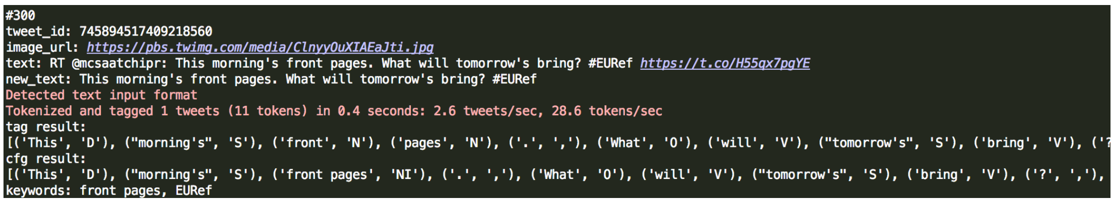
Step 2: Re-tagging using CFG:

This	Morning’s	front	page	.	What	will	Tomorrow’s
D	S	NI		,	O	V	S
bring	?	#EURef					
V	,	#					

Step 3: Spotting the Keywords:

This	Morning’s	front	page	.	What	will	Tomorrow’s
D	S	NI		,	O	V	S
bring	?	#EURef					
V	,	#					

Thus, the keywords are:
{front pages, EURef}
The Console Outputs:



```
#300
tweet_id: 745894517409218560
image_url: https://pbs.twimg.com/media/ClnyyOuXIAEaJti.jpg
text: RT @mcsaatchipr: This morning's front pages. What will tomorrow's bring? #EURef https://t.co/H55qx7pgYE
new_text: This morning's front pages. What will tomorrow's bring? #EURef
Detected text input format
Tokenized and tagged 1 tweets (11 tokens) in 0.4 seconds: 2.6 tweets/sec, 28.6 tokens/sec
tag result:
[('This', 'D'), ("morning's", 'S'), ('front', 'N'), ('pages', 'N'), ('.', ','), ('What', 'O'), ('will', 'V'), ("tomorrow's", 'S'), ('bring', 'V'), ('?
cfg result:
[('This', 'D'), ("morning's", 'S'), ('front pages', 'NI'), ('.', ','), ('What', 'O'), ('will', 'V'), ("tomorrow's", 'S'), ('bring', 'V'), ('?', ','),
keywords: front pages, EURef
```

4.3 Summary

In the keyword extraction part, I did some research on keywords extraction and learnt the two main methods for keyword extraction. At last, I chose the algorithm for short-text. I followed the steps of: tokenizing, tagging, re-tagging with CFG, keyword spotting. And I got good result at last.

5 Evaluation of Our System

5.1 Evaluation Methods

There are several basic methods for evaluation:

5.1.1 Precision

Precision is equal to the rate of number of relevant images in total images.

5.1.2 Mean Average Precision

Mean Average Precision is the Average Precision for each rank for a given search query, average over all queries.

5.1.3 Discount Normalized Cumulative Gain (DNCG)

Gain is score of document's utility. Normalized by rank and cumulative over entire result set to calculate DNCG.
We can give a score of how relevant is the image to the given search term.

5.2 Evaluation Result

We chose 50 sets of recent news as test set. We searched the test cases using our search system and google image for comparison. And we find that our system is good at searching for the images of recent news, especially for those news images with ambiguous text. Because our system can give a good tag for those untagged images with ambiguous text.

6 Conclusion & Future Work

6.1 Conclusion

In this semester, we have built a keyword search system for Twitter images mainly for news topic.
We first get data from Twitter and preprocess the data. Then we use some parsing methods and semantic understanding to get the keywords from each tweet. After getting the keywords, we use knowledge graph to expand the keywords.
As we get all the data we want: the tweet image, the keywords and the expanded keywords, we build the search system using these data.
In the end, we use some methods to evaluate our system. And we find that our system is good at searching for the images of recent news, especially for those news images with ambiguous text.

6.2 Future Work

I have some naïve ideas to optimize the keyword extraction algorithm.
The first is to use CSG (Context-sensitive grammar) to replace the CFG in the re-tagging step in order to get better semantic result.
The second idea is that in the keyword spotting step, we arbitrarily let the words or phrases tagged by 'NI', '^', '#' as the keywords. Is it possible that we can use some machine learning methods to train some models for spotting the keywords? Because machine learning methods are more objective than the subjective way.
The third idea is that we can use some classification or cluster method to put the tweet into certain topic, then extract the keyword based on its topic.

7 References

[1] Brendan O'Connor, Michel Krieger, David Ahn, TweetMotif: *Exploratory Search and Topic Summarization for Twitter*, Proceedings of the Fourth International AAAI Conference on Weblogs and Social Media

[2] Slobodan Beliga, Ana Mestrovic, Sanda Martincic-Ipsic: *An Overview of Graph-Based Keyword Extraction Methods and Approaches*, UDC 004.91

[3] Timonthy Baldwin and Su Nam Kim. 2010. Multi- word expressions. *In Handbook of Natural Language Processing*, Second Edition. CRC Press, Tay- lor and Francis Group.

[4] CMU NLP Group: *A Dependency Parser for Tweets*

[5] CMU NLP Group: *Annotation Guidelines for Twitter Part-of-Speech Tagging Version 0.3 (March 2013)*

[6] CMU NLP Group: *Improved Part-of-Speech Tagging for Online Conversational Text with Word Clusters*

[7] CMU NLP Group: *Part-of-Speech Tagging for Twitter: Annotation, Features, and Experiments*

[8] CMU NLP Group: http://www.ark.cs.cmu.edu/TweetNLP

[9] Ken Barker and Nadia Cornacchia. 2000. Using noun phrase heads to extract document keyphrases. In *Proceedings of the 13th Biennial Conference of the Canadian Society on Computational Studies of Intelligence: Advances in Artificial Intelligence*, pages 40–52.

[10] ZHAO, Xin; JIANG, Jing; HE, Jing; SONG, Yang; ACHANANUPARP, Palakorn; LIM, Ee Peng; and LI, Xiaoming. *Topical Keyphrase Extraction from Twitter.* (2011). Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Portland, Oregon, June 19-24, 2011. 379-388.

[11] Y. Ben Ayed, D. Fohr, J.P. Haton, G. Chollet, *Keyword spotting using support vector machines*, in: Text, Speech and Dialogue, Lecture Notes in Computer Science 2448, 2002, pp. 285–292.

[12] Shima Tabibian, Ahmad Akbaria, Babak Nasersharifc: *A fast hierarchical search algorithm for discriminative keyword spotting*, Information Sciences 336 (2016) 45–59

Appendix 1: Minutes of the Meetings

1st Meeting:
Date: 9 Feb (Thu)
Time: 13:00 – 15:00
Place: LC-08
Content: Introduction to the project & arrange the work for every one

2nd Meeting:
Date: 16 Feb (Thu)
Time: 13:00 – 15:00
Place: LC-03
Content: Do some research on the related works and share the ideas

3rd Meeting:
Date: 23 Feb (Thu)
Time: 13:00 – 15:00
Place: LG1
Content: Beginning of the work

4th Meeting:
Date: 9 Mar (Thu)
Time: 13:00 – 15:00
Place: LC-05
Content: Go on with the project

5th Meeting:
Date: 16 Mar (Thu)
Time: 13:00 – 15:00
Place: Lg1
Content: Focus on the Difficulties in the work

6th Meeting:
Date: 11 May (Thu)
Time: 13:00 – 15:00
Place: Lg3-08
Content: Wrap up the work

7th Meeting:
Date: 20 May (Sat)
Time: 14:00 – 16:00
Place: LC-06
Content: Wrap up all the work

Appendix 2: Grading Criteria

Proposal – 10%
Code & Report – 60%
Final Presentation – 30%