Event summarization from news and tweets correlation

Cavallari Sandro
Giglio Marco
Morettin Paolo

ABSTRACT

In this report we consider the task of extracting a summary of the main event that caused a shift on the opinion of Twitter users. This paper presents several techniques which can be used to analyze these *sentiment shift* and to find the event which caused them. For each technique presented below some aspects are taken in account, which are important in the context of a data mining application, such as scalability and efficiency. A comparison of the different techniques is shown as well.

Introduction

This work aims to find a summary of the events which caused a *sentiment shift*.

While event extraction Natural Language Processing (NLP) techniques are mature, their performance on tweets inevitably degrades due to the inherent sparsity in short texts. Since tweets contain heterogeneous language structures and are at most 140 characters long, to extract event from tweets is quite tricky: instead linking tweet to news articles allows us to extract data from well structured and longer text than tweets.

Moreover NLP document compression techniques usually exploits language dependent methods, whereas we want a methodology as language independent as possible.

This paper is structured as follows: in the next chapter we present some works which are somehow related with the problem we are addressing, then we present a more formal definition of our problem and some techniques which might be used to solve it. The techniques which will be later discussed are:

- a technique exploiting the *SpaceSaving* algorithm presented in [8];
- a technique exploiting *Latent Semantic Indexing* [1];
- the third one based on the popular *Tf/Idf*;
- the last (but not least) which take advantage of ngram graphs [4]

We will discuss each of these showing their advantages and disadvantages, their efficiency and scalability issues. In addition, we will show experimental results in order to compare their results and performances. At the end of the paper we will present some conclusions and draw some suggestions on how this work may be extended.

Related Work

We can distinguish two main subgoals in our project:

- compute the correlation between tweet and news
- create a summary of the main event that caused the sentiment shift

There are several works, in literature, which refer to similar tasks. In particular, due to Twitter's increasing popularity, in the last years there have been several works regarding the correlation and analysis of the data. Weiwei Guo et al. [5] proposed a framework to link tweets and news and to extract from the resulting correlation missing aspects of the tweet-event. In their research, rather than using a LSI technique, they propose a methodologies based on the Weighted Textual Matrix Factorization [6] [WTMF] model(the 2012 de facto standard). This approach revealed to return good correlation results, making WTMF a strong tool for baseline creation.

Some works which are complementary with respect to our might be found in the field of recommendation systems; in particular systems which address the problem of recommending news given some user's features. However, there is an important difference between this task and our, since tweets are much more heterogeneous and characterized by a much smaller length than news.

Recently *Google* presented a system for *multi-sentence compression* (MSC) [7], which aims to compress several senteces trying to mantain high readability and to not lose the semantic content of the text. Their approach explits word gram graphs, combining them with some semantic data. The combination of the two approaches allows redundancy removal while mantaining a readable sentence as output.

Problem Definition

We define *sentiment* the measurement of feelings performed using specific tools analyzing the stream of tweets. This kind of measurement is usually done using some supervised machine learning techniques together with NLP techniques.

Tools exist to track the *sentiment* toward a certain topic and to identify changes in it. In particular we can consider three major events in a sentiment timeline:

1. there is a change of sentiment from positive to negative (or viceversa)
2. there is no change of sentiment, but there is a peek in the graph describing it or its value crosses the average value
3. there is no change but in the volume of tweets/second

We define the first of these events as *sentiment shift* and the temporal window in which it happens is called *contradiction point*[CP] or *contradiction window*.

In addition we define *contradiction tweets* and *contradiction news* respectively the tweets and news published inside a contradiction window and *background tweets* and *background news* the set of all tweets and news regarding the addressed topic.

Given these definitions, the problem we aim to solve is to find, among the contradiction news, the ones which are the more representative of the contradiction tweets and to present to the user a summary which describes them.

Proposed Approach

In this section we present different approaches which might be used to solve the problem we are addressing. Of course, each approach has advantages and disadvantages which we will list while showing them.

SpaceSaving approach

Description of the algorithm

SpaceSaving [8] is an algorithm which was first presented by Metwally et al. in 2005 to efficiently compute the most frequent terms in a data stream. It allows the user to find k words which are among the most frequent in the given stream. Although this is a heuristic algorithm whose result's correctness is not guaranteed, SpaceSaving is able to specify the upper bound of the error for each word presented as output.

The classic implementation is based on a fixed size list of tuple

$$(word, occurrencies, error)$$

The stream is read word by word and at any time one of the following condition is verified:

1. the word is already in the list of frequent terms
2. the word is not among the frequent terms, but the list is not full
3. the word is not among the frequent terms and the list is full

If the first condition is verified, then the algorithm will proceed incrementing the number of occurrencies of that term; if condition 2 is verified, instead, we add that word to the list of frequent terms with number of occurrencies 1 and number of errors 0; if condition 3 is verified, then we scan the list for the term with the lowest number of occurrencies and replace it with the new word. When this replacement takes place we set the number of errors equal to the number of occurrencies of the word we just replaced, then we increment the number of occurrencies.

Our python implementation of this algorithm is slightly different from the classic one described above, since we test the word against a list of stop words and we added a hashmap to avoid useless replacement. In our implementation, then, the following steps are performed:

1. if the word is in the stop word lists, does nothing
2. otherwise compute the hash of the given word
3. increment the value associated to that hash
4. check whether the value stored in the hash map is above the number of occurrencies of the less frequent term tracked by the SpaceSaving algorithm. If so, the replacement occurs, otherwise nothing happens.

The output of this algorithm is a couple $(word, value)$ where the value is computed as

$$value = occurrencies - error$$

In order to reduce the noise we chose to use only couples where the value is above a certain threshold.

Description of the methodology

We have seen how we exploit the SpaceSaving algorithm to obtain the list of frequent terms in a text and their values.

To accomplish our task, we ran the program described above on the set of tweets inside a *contradiction point*, thus obtaining the list of frequent terms within the contradiction; then, we read all the news which were published in the same time period (a certain error between the publishing date and the contradiction window) and for each of them we compute its score as

$$newsScore = \sum_{w \in W} value(w)$$

where W is the list of words in the news and the value is null if that term is not in the *frequent term list*.

The news with the highest score is selected as the one causing the shift and its first words are presented as output to the user.

This approach revealed to be very fast and from its formulation we can see that it can be applied incrementally. As a drawback, we must say that it does not take in account the background coming from the topic, hence it is not able to discern whether a given frequent term is informative or not (e.g. running this program on the Obama topic it is likely that the most frequent terms will be “President”, ‘Barack”, “Obama”, “USA”, hence a news containing more repetitions of these words will have a high score, even if those words are not informative at all)

LSI

We used a bag-of-words approach exploiting a technique called Latent Semantic Analysis (also known as Latent Semantic Indexing), formalized by Scott Deerwester et al. in 1990 [1]. LSA is an information retrieval statistical approach, that can be used to find semantical similarities between sets of documents. It takes advantage of Singular Value Decomposition on a vector space of term frequencies in order to create, given a set of documents, an *index* matrix, that can be queried with other documents to find the most similar one. In order to find correlations between the sentiment shifts and the news articles, our approach works as follows:

- All the tweets corresponding to a CP are merged in a single document.
- Every document is preprocessed and converted in the bag-of-word representation.
- The LSA model is trained on the whole set of news for the given topic.
- The set of candidate news to be tested is chosen according to a window, and used to create an index.
- The news are sorted in order of similarity, optionally weighted by a factor considering the time distance of the news from the CP.
- All the candidate news over a certain threshold are considered to be correlated and a set of words calculated with TF-IDF is returned.

Preprocessing

During the preprocessing, every document is filtered and converted from a string to a bag-of-word representation. The following steps are done in this phase:

1. The punctuation characters are substituted with a whitespace.
2. The string is tokenized.
3. Stopwords and tokens appearing less than t times in the document are removed.

LSA and candidate news selection

Once the documents are filtered and transformed in a bag-of-words representation, a dictionary is created containing the association between tokens and a positive integer identifier. This is done in order to convert on-the-fly each document D in a vector of tuples $(i, n(D, i))$ where $n(D, i)$ is the number of occurrences of the token i in the document D. The LSA model is then trained with all the news available for the given topic. Given a sentiment shift over a time interval $[T_b, T_e]$, the news can be selected using a fixed size window $[T_b - c, T_e]$ or a more sophisticated approach that considers the density of the tweets inside the shift. The latter seems reasonable since news that correlates with a shift are likely to be near the shift the more the *hype* increases.

News scoring and summarization

With the candidate news a matrix index is then created and the document representing the shift is compared. This operation produces a vector of similarities of the shift with the candidate news. These scores do not take into account the hype and the temporal distance of the news w.r.t. the sentiment shift. Unfortunately, we didn’t had time to investigate this aspect and find a reasonable weighting function on those parameters. The candidate news passing a threshold are considered to be correlated with the shift, so for each of them the list of the k words with higher TF-IDF values are returned.

Implementation

The methodology described above was implemented in python 2.7 using a library called gensim [3]. It was possible to store all the preprocessed data for a single topic on main memory and perform all the computation in a single pass. The choice of the number of dimensions of the LSA space is non-trivial. For small dataset like ours, a number between 50 and 100 has proven to be optimal [2]. At the same time, for a more specific discrimination in an homogeneous topic a larger number is suggested. For those reasons we used a 200 dimensions vectorial space.

N-Gram Graph

N-Gram Graph (NGG) is a NPL tool initially proposed by George Giannakopoulos [4] that uses word or character n-grams in order to achieve documents summarization. The NGG tool basically slices the text in word or character n-grams and then represent them in a graph $G = \{N, E, L, W\}$ according to the following structure:

- N is the set of nodes created for every different n-gram in the text
- E represent the edge of the graph; two nodes are connected if they are “close” or whithin a *distance window* from each other
- L is a labelling function which assigns labels to every node and every edge (define the size of the n-gram)
- W is the weight function which assigns weights to every edge according to the number of times that two n-gram appear close one to the other

The big advantage coming from the use of this methodology is language independence, since it makes no assumption on the underlying languages and allows text manipulation trough graph operations.

In particular two operation are necessary for our goal:

- the **Intersection** operator between two graphs G_1 and G_2: which returns a resulting graph with only the common edges of G_1 and G_2 averaging the weights of the original edges assigned as the new edge weights(example in figure 1)

- the **Normalized Value Similarity**[NVS] function that for every n-gram rank, indicating how many of the edges contained in graph G_i are also contained in graph G_j, considering also the weights of the matching edges and normalize the result respect the graph size

In particular $NVS(G_i, G_j) = \frac{VS(G_i,G_j)}{SS(G_i,G_j)}$ where:

$$VS(G_i, G_j) = \frac{\sum e \in G_i \frac{\min(w_e^i, w_e^j)}{\max(w_e^i, w_e^j)}}{\max(\mid G_i \mid, \mid G_j \mid)} \quad (1)$$

$$SS(G_i, G_j) = \frac{\min(\mid G_i \mid, \mid G_j \mid)}{\max(\mid G_i \mid, \mid G_j \mid)} \quad (2)$$

Since NGG tools allow to compute both the tweet-news correlation and summary creation, this methodology is split in different section.

Tweet-News Correlation

To compute the correlation between contradiction tweet and news, this methodology use the base idea of exploiting the NVS similarity function as a correlation function: higher the similarity of the tweets n-gram graph representation and the news n-gram representation, higher will be the correlation.

More in detail this procedure perform the following step:

- Merge all the tweets corresponding to a CP are in a single document n-gram graph representation (G_{tw}).

- Compute the news n-gram graph for the news that fall inside a time slots defined by the contradiction point windows increased on both side by a time windows duration (G_n)

- for the news that are inside the time slot, the NVS is computed and represent the correlation value between news and tweet $G_{sim} = NVS(G_n, G_{tw})$

The usage of the time windows is needed given that NVS similarity values have no time domain impact, and with out the time windows isn't possible to discharge news that are distant in time form the correlation computation.

This approach require a lot of computation, since have to compute the graph representation of all the tweet and for every news inside the time windows compare the tweet with the news text. Instead NGG allow to compute the similarity between text without the usage of grammar information and use this similarity as correlation value: even if has no time relation.

Summary Creation

The second goal of this research is to create a summary of the event that cause the sentiment shift.

For this purpose we use the news with the highest correlation value obtained at the step before. Since the goal is to create a summary of 150 word, we suggest to take in consideration only the best 4 news. Increasing the summary length can require more news.

The summary algorithm perform the following action:

- sort the news according the correlation value obtained before
- save all the sentence of text best 4 news in a set; the sentence detection is perform using OpenNLP
- create the intersection graph of the best 4 news (G_{bn})
- for every sentence saved compute the NVS between the sentence n-gram graph and the intersection graph ($NVS(G_s, G_{bn})$)
- create the summary using the sentence with the highest NVS value until we reach 150 word

this algorithm create a summary composed by the most significant sentence of news with the highest correlation value using a back-loop approach, but is not able to remove redundant sentence. Even this, a summary composed by sentences result to be more human readable than a list of key word.

Experimental Evaluation

Experimental setup

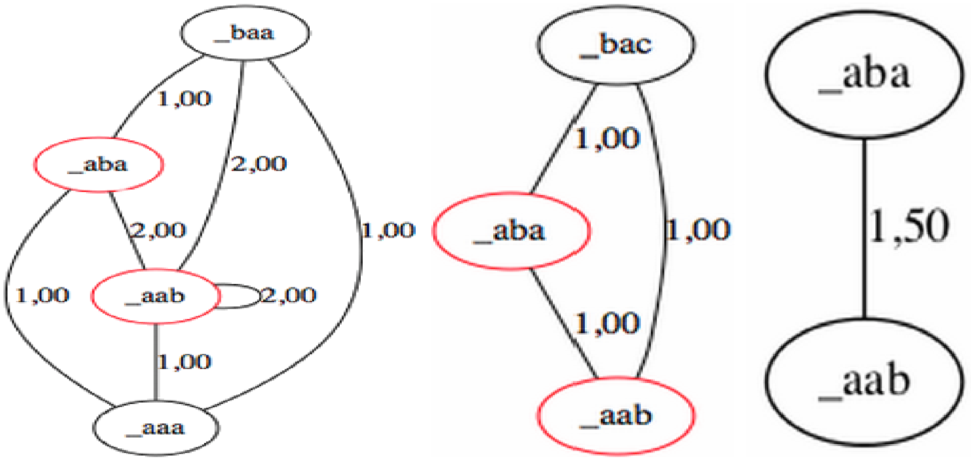
In order to perform the experimental evaluation we used a tool to automatically detect sentiment shift in tweets coming from year 2009 and regarding several topics. We also downloaded news from the *New York Times* and *ABC Australia* on the same topic and spanning on the same period of the tweets.

It's important to remember that tweet have different language structure that need to be normalized, so for cleaning purpose, the following operations were performed on the tweet text before starting the computation of all the different methodologies:

- URLs removal from tweets using a regular expression
- conversion from Unicode to ASCII

In addition, while parsing the news, we considered the possibility that the opinion expressed by tweets might be both delayed or in advance with respect to news (e.g. if a movie becomes popular it is probable that news spread very fast on twitter, but slower on newspaper, whereas other topics are discussed on twitter only after news about them came out for several days). Thus, we considered a enlarged time window for news which starts 5 days before the beginning of the contradiction and last up to 5 days after the end of it.

After that, we manually labelled each contradiction point with the event which caused it. The topics, contradiction points and events used for the experiments are showed in table 1



(a) G1="aaabaab" (b) G2="aabac" (c) Intersection Result

Figure 1: An example of the intersection operation

Contr. point	Contr. window	Event
Cern1	2009-10-29 - 2009-11-08	On November 3rd a bird drops a piece of bread which cause LHC overheating.
Cern2	2009-11-24 - 2009-12-04	On November 30th LHC accelerates protons to an energy of 1.18 TeV, becoming the world most powerful energy particle accelerator
FortHood	2009-11-02 - 2009-11-06	On November 5th a US marine kills 13 people
HangOver	2009-06-21 - 2009-06-27	undetermined or movie released on June 5th
Lcross	2009-10-31 - 2009-11-08	Preliminary findings from Lcross. Others announced
Jackson1	2009-06-19 - 2009-06-25	On June 25th Michael Jackson dies
Jackson2	2009-08-23 - 2009-08-29	On august 28th another popular musician, Adam Goldstein, dies. The day after, August 29th is Jackson's birthday
SwineFlu	2009-06-19 - 2009-06-23	The swine flu is recognized as a pandemic

Table 1: Contradiction points used for experimental evaluation

Experiments

SpaceSaving evaluation

The results achieved using the *SpaceSaving* methodology with a maximum error of five days on the contradiction window are shown in table 2.

Contr. point: Cern1	**Label:** bird accident	**Number of news:** 3
Score mean: 1124.33	**Score std. dev:** 364.31	**News score:** 1545
Result: Peckish bird crashes atom smasher. A peckish bird has briefly knocked out part of the world's biggest atom smasher by causing a chain reaction with a pi…		
Frequent terms: lhc(442), cern(166), bird(114), baguette(71), bread(65), down(59), large(47), hadron(47), collider(46), dropped(43)		
Contr. point: Cern2	**Label:** power on and record	**Number of news:** 6
Score mean: 6014.83	**Score std. dev:** 3658.54	**News score:** 10386
Result: Atom-smasher sets energy record. The world's biggest atom-smasher has set a world record by accelerating to energy levels that had never been previously…		
Frequent terms: lhc(1360), cern(653), record(320), world(251), beam(206), beams(192), energy(188), collider(178), tev(153), first(148)		
Contr. point: Fort Hood	**Label:** shooting	**Number of news:** 78
Score mean: 36016.0	**Score std. dev:** 51063.86	**News score:** 323404
Result: Updates on the Shootings at Fort Hood. On Friday, The Lede is providing updates on the aftermath of Thursday's deadly shooting spree at Fort Hood, the T…		
Frequent terms: hood(3889), fort(3839), prayers(693), families(658), shooting(631), out(506), texas(478), army(442), thoughts(427), dead(406)		
Contr. point: Hangover	**Label:** movie released	**Number of news:** 5
Score mean: 2465.0	**Score std. dev:** 1622.42	**News score:** 4081
Result: Despite Jeers From Critics, Sequel Scores A Huge WinHorrid reviews couldn't dent "Transformers: Revenge of the Fallen," which demonstrated once agai…		
Frequent terms: hangover(2837), watch(459), see(439), free(374), movie(301), funny(231), go(213), saw(196), good(172), going(162)		
Contr. point: Lcross	**Label:** preliminary findings	**Number of news:** 0
Score mean:	**Score std. dev:** 0	**News score:** 0
Result: *No news in the selected time period*		
Frequent terms: lcross(12), moon(7), nasa(4), impact(4), data(3), lunar(2), crater(2), full(2), story(2), mercury(2)		
Contr. point: Jackson1	**Label:** Jackson's death	**Number of news:** 118
Score mean: 4002.11	**Score std. dev:** 4130.64	**News score:** 38462
Result: Michael Jackson, 50, Is Dead. This post is written by Jon Pareles, Ben Sisario and Brian Stelter in New York and Brooks Barnes in Los Angeles.		
Frequent terms: jackson(511), michael(505), think(31), video(28), bad(27), twitter(24), make(23), listening(22), out(22), follow(22)		
Contr. point: Jackson2	**Label:** Jackson birthday, Goldstein's death	**Number of news:** 59
Score mean: 254620.51	**Score std. dev:** 213818.49	**News score:** 868755
Result: Court Papers Say Lethal Levels of Anesthetic Caused Jackson's Death. Lethal levels of a powerful anesthetic caused Michael Jackson's death, according to…		
Frequent terms: jackson(45879), michael(45655), birthday(8430), happy(6969), death(4198), homicide(3165), love(3089), mj(2599), video(2145), day(2090)		
Contr. point: SwineFlu	**Label:** pandemic	**Number of news:** 35
Score mean: 1091.66	**Score std. dev:** 1043.49	**News score:** 5763
Result: In New Theory, Swine Flu Started in Asia, Not Mexico. Contrary to the popular assumption that the new swine flu pandemic arose on factory farms in Mexic…		
Frequent terms: swineflu(348), flu(178), swine(150), tags(73), welcome(40), cases(40), everybody(37), influenza(32), /(27), visitors(23)		

Table 2: Results achieved using SpaceSaving

LSI evaluation

Tf/idf evaluation

Ngram Graph evaluation

As for the other methods for the NGG computation we used a time windows of five days and in table **??** are reported the obtained results

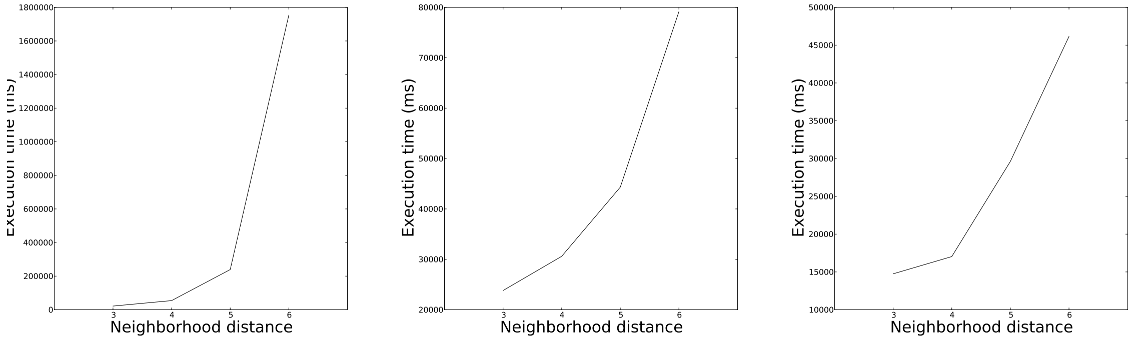
All the experiment about Ngram graph where conduced on a 2,4 GHz Intel Core 2 Duo with 4 GB 1067 MHz DDR3 RAM memory. All the code was compiled with intellij IDEA on Java 1.8 language. The reported test where conduced on Cern topic data set, since is the one that have multiple contradiction points and allowed us to test the methodology changing the following parameter:

- rank of the graph
- neighbourhood distance[ND]
- time windows

In figure 2 is reported the computational time respect the neighbourhood distance for different rank values. It is easy to notice how for all the rank there is an exponential behaviour respect to the neighbourhood distance. Increasing ND will increase exponentially the number of edges that a graph contains, so the computation will became exponentially longer. Since neighbourhood distance strongly affect the computational time and have no big difference on the summary obtained, is reasonable to use a ND between 3 and 4.

In figure 3 is possible to see how the computation time change respect to the rank value at different neighbourhood distance. Surprisingly for a neighbourhood distance = 3 the computation time with a n-gram graph of rank=3 is slightly higher respect to a rank=2. That is quite strange since slicing a text in two-gram should produce more node that slicing a text in three-grams. Probably with rank = 2 and ND = 3; near grams are part of the same word creating graph easy to compute. This scenario produce a smaller computation time in a two-gram graph that a three-gram graph where we have more distinct node and so edge to check.

Is possible to notice how for bigger ND values, the computation time for of the 2-gram diverge, instead 3-gram became more similar at the 4-gram behaviour. Probably because computing the similarity between 2-gram graph is an operation quite expensive since they have many edges to check.

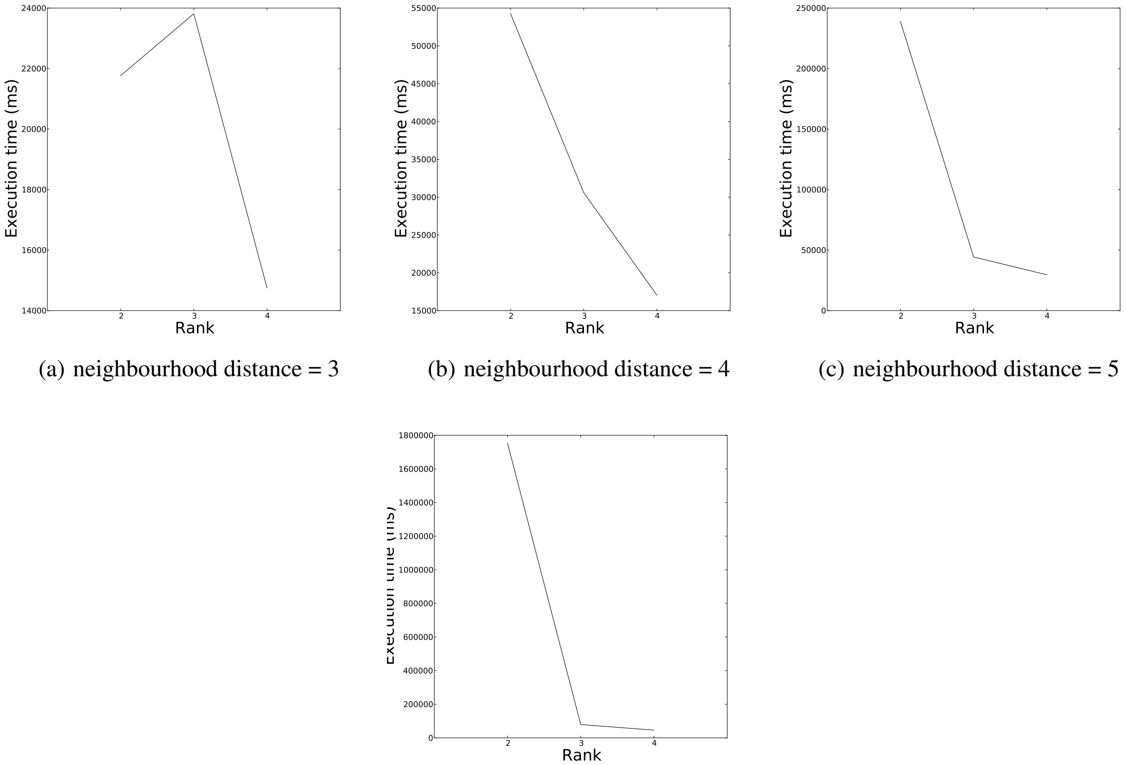


(a) rank = 2

(b) rank = 3

(c) rank = 4

Figure 2: Execution time for different rank value



(a) neighbourhood distance = 3

(b) neighbourhood distance = 4

(c) neighbourhood distance = 5

(d) neighbourhood distance = 6

Figure 3: Execution time for different neighbourhood distance

Conclusions

Future improvement

Redundancy removal

NGG is a technique developed primary with the goal of *inter-summary* generation: a summarization process that takes into account information already available to the reader. In order to achieve this goal, in a good summary every new sentence must add as less redundant information as possible. According to the proposal of the NGG developer, implementing the following function can improve the quality of the summary created:

- Starting form the first sentence of the summary, compute his NGG representation G_{sum}
- Remote G_{sum} form the intersection of all the news to summarize (C_u). The new graph G'_{sum} = $G_{sum} \triangle G_{in}$
- For every candidate sentence of the news that has not been already used:
  - extract its n-gram graph representation G_{cs}
  - keep only $G'_{cs} = G_{cs} \triangle C_{in}$, because we want to add sentence with low redundancy
  - Computing the NVS between G'_{cs} and G'_{sum} give a *redundancy score*
- rank the candidate sentence using as score their NVS value respect G_{in} decreased by the redundancy score

this functionality was developed by the author of NGG tool and was used for inter-summary generation. Unfortunately for time reason we weren't able to test his performance in our research, but we think that will produce much more human readable summary with more information.

Adaptive time windows

All the methodologies compute the correlation between news and tweet using a fixed-length time windows. This approach is quite simple and can produce good performance since there are a good number of news in the selected time slice.

In order to face tricky situation where there are only few news for the selected interval, a more sophisticated approach should adapt the length of the time windows according to the number of news available: precisely, more news are present in the time period of the sentiment shift and shorter should be the amount of time added to the time windows of CP. Developing and testing an adaptive time windows can increase the performances of the all the methodologies and is one of the main point for the future improvements.

1. REFERENCES

[1] Indexing by Latent Semantic Analysis, Scott Deerwester , Susan T. Dumais*, George W. Furnas, Thomas K. Landauer and Richard Harshman

[2] LANDAUER, Thomas K., et al. (ed.). "*Handbook of latent semantic analysis*". Psychology Press, 2013.

[3] ŘEHŮŘEK, Radim, et al. "*Software framework for topic modelling with large corpora*". 2010.

[4] Automatic Summarization from Multiple Documents : N-Gram Graph, George Giannakopoulos and Ncsr Demokritos, 2009

[5] Linking Tweets to News: A Framework to Enrich Short Text Data in Social Media, Weiwei Guo, Hao Li, Heng Ji and Mona Diab

[6] Modeling Sentences in the Latent Space, Weiwei Guo and Mona Diab , 2012

[7] Multi-Sentence Compression: Finding Shortest Paths in Word Graphs, Katja Filippova

[8] Metwally A. et al, "*Efficient Computation of Frequent and Top-k Elements in Data Streams*", Lecture Notes in Computer Science Volume 3363, 2005, pp 398-412