Research Paper in Business Analytics

Identifying News Articles in Dutch Tweets

Nikita Galinkin

supervised by
Marijn ten Thij

Vrije Universiteit Amsterdam
Faculty of Sciences
Business Analytics
De Boelelaan 1081a
1081 HV Amsterdam

October 2, 2016

Abstract

Twitter is a microblogging platform that gives people fast and easy real time access to information on what is happening in the world. In this work we aim to identify the tweets belonging to a given news article. We use a collection of dutch tweets and articles from the news portals nu.nl and nos.nl. Using a vector space model (VSM) with term frequency inverse document frequency (tf-idf) weighting and cosine similarity to compare each tweet to a collection of news articles. Time windows are introduced to deal with the small likelihood of a tweet belonging to a news article that was published more than a certain time before or after the tweet. The time window of four hours proves to be most successful with an overall accuracy of 90.82% and an accuracy of 57.65% for tweets that do not contain the URL. The results of this research allow to study the spread and speed of news in future works.

Contents

1 Introduction

When in 2009 the US Airways flight 1549 crashed into New York's Hudson river the first photo from the scene was shared on Twitter within minutes and beat traditional news broadcasters by around 15 minutes to the story. The picture immediately went viral triggering a snow ball effect and causing Twitter's picture sharing service to crash. This moment is one of the milestones in Twitter history, marking the beginning of a new way news is brought to the world.

The study conducted by Kwak et al.(2010) [8] has shown that nowadays over 85% of trending topics in Twitter are news. Rosenstiel et al.(2015) [9] found out that over 85% of the Twitter users use it for news. Twitter's main appeals are its real time nature, diversity of sources and simplicity of scanning. People can follow events in real time getting updates faster than on any news site and they can do so on the fly no matter whether they are rushing from meeting to meeting, sitting in a metro or chilling at a pool. Thanks to mobile technologies Twitter is accessible any time anywhere.

1.1 About Twitter

Twitter is a microblogging platform that engages monthly up to 313 Million users (as of Q2 2016) supporting them in 40+ languages. Twitter allows registered users to share pictures and little text messages of up to 140 characters, called *tweets*, with their followers. A *follower* is a person who subscribed to see your tweets. Follower relationships in Twitter are not reciprocal, the user you follow does not need to follow you back. Users have the possibility to *retweet* a tweet to show it to their followers or to directly respond to a tweet via *reply*. Other users are addressed using the @ sign in front of a username and keywords are highlighted through a # sign in front of them to make the tweet easy to find in Twitter searches, in the Twitter world they are called *hashtags*.

1.2 Goal of this Research Paper

The initial goal of this research was to find out how fast and how far news spread in the dutch tweetosphere and what factors influence that. In order to investigate those questions, information on which tweet discusses which news is needed. This is no trivial task, since tweets are not labeled and we do not know which tweet belongs to which news a priori. One possibility is to use hashtags of buzzwords from several news articles. But the problem is that in our data set only 30% of the tweets contain hashtags, so using this method will lead to a great loss of information. Another possibility is to look at tweets that contain a direct URL of an article, but in our data set far less than 1% of the tweets would be usable. Additionally going with the tweets containing a URL is also coming with a huge loss of valuable information, since most people do not attach a URL every time they discuss a topic on Twitter.

Thus the initial goal has been amended and the new goal is to automatically categorize the tweets according to the news articles they belong to. In this work we will use news articles from two of the most popular news sites in the Netherlands - nu.nl and nos.nl.

With this research we are creating an algorithm that is capable of identifying a news topic in a tweet and associating the tweet with a given news article.

1.3 Structure of this Research Paper

The paper is structured as follows. Section 2 discusses relevant literature on the topic. The data set used in this research is described in section 3. Section 4 gives the theoretical framework used while creating the algorithm. Results are discussed in section 5. Lastly we conclude the the findings in section 6.

2 Related Work

In this section we give an overview over interesting research related to the works of this research paper. The first works are about Twitter itself and the way it is used. The latter articles study event or news in Twitter, their recognition, spread and forecast of popularity.

A first large quantitative study on Twitter was conducted by Kwak et al.(2010) [8]. The authors compare Twitter to a human social network and notice deviations from known characteristics of the latter one. A ranking of users by number of followers and by PageRank is found to be similar. The authors also classify trending topics and find out that over 85% of topics are news. Another interesting finding of this work is that any retweeted tweet is to reach an average of 1.000 users no matter the number of followers of the original tweet is.

A study by Rosenstiel et al.(2015) [9] reveals that nearly 9 in 10 Twitter users use Twitter for news and out of those 74% do it daily. On the reason why people use Twitter for news the most chosen answers are that it allows them to access news immediately in real time, that it allows them to come across sources they normally would not find and that Twitter is easy to scan. This is an interesting insight into the reasons people use Twitter for news.

Bhattacharya et al.(2012) [3] study the spread of news articles from different sources, their survival rate and life span. For their study they use tweets that contain a URL of the news. This is an interesting work that gives further insights into the topic.

In [5] Hansen et al.(2011) study what affects virality in Twitter. They conclude that tweets about news are more likely to be shared if they have a negative sentiment.

Using classifiers Bandari et al.(2012) [1] forecast the popularity of a news article in social media prior to their release. They are able to build an algorithm that predicts ranges of popularity on Twitter with an overall 84% accuracy. Another interesting thing they note is that "popular news threads take about 4 days until their popularity starts to plateau". This is close to one of our findings that will be discussed and used in this work.

Becket et al.(2011) [2] explore approaches for real-world event detection in streams of Twitter messages. The main finding of their work is that support vector machines perform this task best with an average precision of over 70%.

Kunneman et al.(2014) [7] also worked on the detection of events in tweets. Their approach is based on term-pivot clustering and reaches a precision of roughly 80%.

Sakaki et al.(2010) [10] analyze tweets to detect earthquakes in Japan. They use semantic analysis classifying tweets into a positive or negative class to understand whether they are relevant. Every Twitter user is used as a sensor and the tweet as sensor data. This enables the application of various methods related to sensory information. With location estimation methods they are able to predict the trajectory of an earthquake or a typhoon. The authors are able to detect 96% of earthquakes with a magnitude of 3 or higher in the studied area.

Hughes et al.(2009) [6] and Chew et al.(2010) [4] analyze the spread of news about emergency events and catastrophes via Twitter.

Combining various artificial intelligence, information retrieval, computational linguistics, and natural language processing methods Vikre and Wold (2015) [12] identify news topics in tweets, cluster them based on similarity and time and find the most representative tweet for the topic using a tf-idf centroid approach. This work is probably the most similar to our work as it also tries to find news in tweets. Instead of comparing the tweets to news article texts Vikre and Wold use natural language processing systems to do so.

3 Data Collection, Preparation and Exploration

3.1 Data Collection

The data collection is performed the same way as in [11]. The tweets are scraped using the filter stream of the Twitter Application Programming Interface (API). As in [11] four streams are set up, although only two are used: the "NL General" and the "NL Specific" streams. The "NL General" stream consists of tweets that contain at least one word from a list of 130 typical Dutch words (e.g., *'een, het, ik, niet, maar, heb, jij, nog, bij'*). The "NL Specific" stream filters tweets according to their author who must be from a list of 1,303 Dutch users (e.g. *@NUnl, @NOS*) and contain terms from a list of 395 entries (e.g., *'brandweer, politie, gewond,ambulance'*). Those two streams give us a broad sample of the desired Dutch tweets. Note that the same tweet might be included in both streams.

The compressed files with tweets from all four streams have a size of 143 GB consisting of 720 files - one for every hour of the month. The size of the decompressed files is about 972 GB.

The decompressed output files are JSON files containing the tweets with all the fields that belong to the tweet (e.g., *id, text, user_id*, ...for a more detailed description see the Twitter API description [1]).

3.2 Data Preparation and Exploration

The data collection process from the "NL General" and "NL Specific" streams allows to gather a data set of 28,550,510 tweets. This is a very large amount that takes a long time to process and is difficult to work with, thus the data preparation step is preceding the data exploration step.

Not all tweets are relevant for our purpose, so the tweets are filtered leaving only the ones that contain the string "nu.nl", "nos.nl", "@NUnl" or "@NOS" in the *text* or the *URL* field or that are in reply to such tweets. We check both fields since the field *URL* usually contains the long versions of a shortened URL. The result of the filtering are 238,010 tweets that make up the final data set. Approximately 90% of the tweets contain a URL.

The attributes extracted from the tweets are the following:

[1]https://dev.twitter.com/overview/api/tweets

• *id_str*	-	id of the tweet
• *text*	-	text of the tweet
• *urls*	-	URLs contained in tweet (their position, short version and expanded version)
• *created_at*	-	date when tweet was created
• *in_reply_to_status_id*	-	id of the tweet that this tweet is in reply to

Having found the ids of all the news articles from nos.nl and nu.nl mentioned in the tweets a new data set is created containing:

- the *id* of the news article
- the *article* text itself
- the *date* the article was published
- the *frequency* of mentions
- the date of the *first mention* of the article in tweets
- the difference between *first mention* and *date*
- the date of the *last mention* of the article in tweets
- the difference between *last mention* and *date*

The attribute *newsid* was added to the Twitter data set.

News articles are downloaded from the HTML site using the Python libraries *urllib.request* [2] and *BeautifulSoup* [3].

Overall 9195 different articles are mentioned in our Twitter data set. 4030 nos.nl articles and 5165 nu.nl articles. 59% of the articles are mentioned 10 times or less.

99% of the articles are first mentioned two hours or less before the article was published. Figure 1 shows the distribution of the days until the last mention of a news article in the tweets. 70% of the articles are last mentioned 2.8 days after publishing, 80% of the articles a little over 4 days and 90% of the articles are mentioned last after almost 8 days. This picture agrees with the findings in [3] and [1]. Figure 2 shows the same picture for tweets not containing a URL. Here 90% of the articles are mentioned at most one day after the article has been published.

[2]https://docs.python.org/3/library/urllib.request.html
[3]https://www.crummy.com/software/BeautifulSoup/

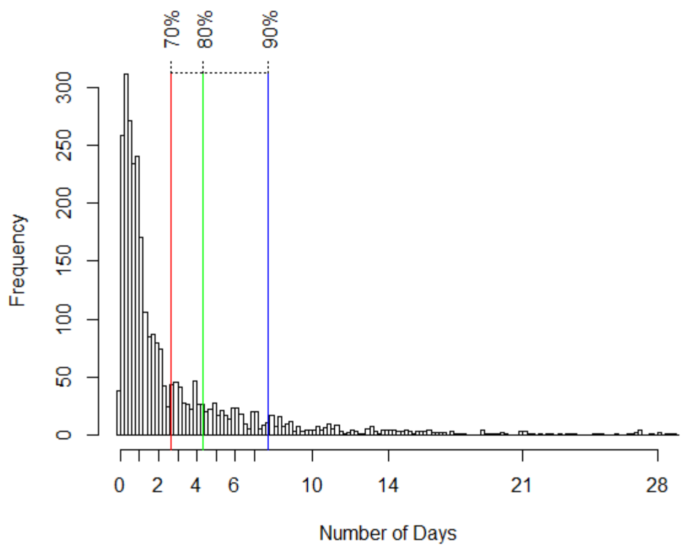

Figure 1: Number of days from article publishing until the last mention in tweets.

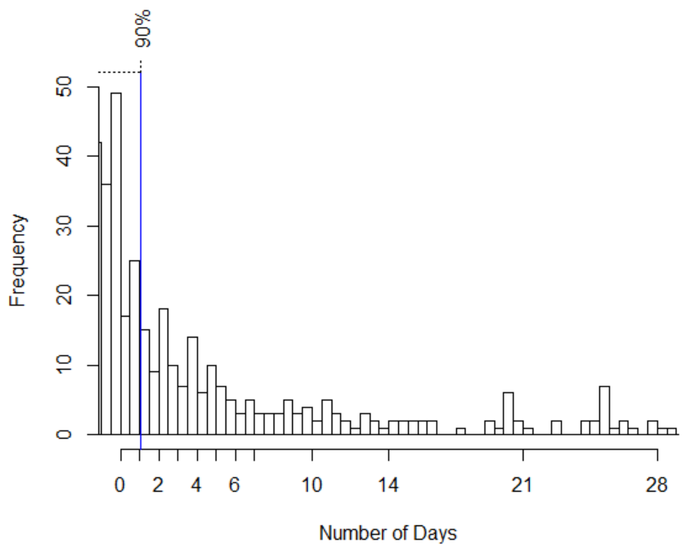

Figure 2: Number of days from article publishing until the last mention in tweets not containing a URL.

4 Theoretical Framework

Having gathered three data sets - the tweets, the nos.nl articles and the nu.nl articles our goal is to find out whether a tweet belongs to a news article and if so to which one.

First we clear up some terminology. Each text is called a *document* and all together they build up a *corpus.* So a *corpus* is a collection of *documents.* Transferring it to the data sets at our hand every separate tweet or news article is a document and the collection as a whole is the corpus.

We use the technique of vector space models (VSM) with term frequency-inverse document frequency (tf-idf) weighting and cosine similarity as the similarity measure. This technique has been observed in several other works on this topic and has proven to be simple but very well performing. The main advantage of this technique is that it does not require any knowledge of the semantics, all that is needed are term occurrence frequencies. In this section the idea of vector space models is first described. Then comes a brief description of the tasks of stemming and stop word removal. Finally the process of calculating the tf-idf and the cosine similarity is explained.

To make the theory easier to understand this section is accompanied by the following example corpus consisting of four documents:

- d_1: "the sun shines bright"
- d_2: "the sky is blue"
- d_3: "The sun in the sky is brighter."
- d_4: "We can see the shining sun, the bright sun"

Let us assume that $d1$ is the tweet and $d2$, $d3$ and $d4$ are the news articles and the goals is to identify to which of the three documents the tweet is most similar.

4.1 Vector Space Model (VSM)

The idea of vector space models is to translate documents from strings into vectors containing their weights so we can apply linear algebra.

This is done in the following way. First the documents are cleared through stemming and stop word removal. Then a dictionary of all the terms that appear in the corpus is created. Now every document can be translated into a vector. The dimension of the vector is the same as the number of unique terms in the corpus. Every entry of the vector is the weight of the corresponding term, in our case it will be the tf-idf of the term.

After that the algorithm does a normalization of every vector through the division by its Euclidean length.

$$\frac{A}{\parallel A \parallel} = \frac{A}{\sqrt{\sum\limits_{i=1}^{n} a_i^2}} \tag{1}$$

This way every vector is scaled down to length 1. This step is done for speeding up the algorithm as it will simplify the calculation of the cosine similarity.

4.2 Stemming

Before starting to count the word frequencies it has to be made sure that all the words with the same root are really the same throughout the different documents. For example when counting the two words "beautiful" and "beauty" they both mean the same, but the algorithm only recognizes the word as a string, not knowing the meaning. So for the computer "beautiful" $\neq$ "beauty". This is why stemming is used - the process of reducing words to their stem. In the example the stemmed version of both words is "beauti". Although it might be that different stemming algorithms produce different stems this is not a problem as long as it is consistent throughout one closed work.

The stemming algorithm used in this research is the *SnowballStemmer* [4], part of the *nltk* [5] toolkit. The *SnowballStemmer* was used because it is available in Dutch and 14 other languages.

In the example the only two words that need to be stemmed are "shines" in $d1$ and "shining" in $d4$. So the stemmed corpus is:

- d_1: "the sun shine bright"
- d_2: "the sky is blue"
- d_3: "The sun in the sky is bright."
- d_4: "We can see the shine sun, the bright sun"

[4]https://pypi.python.org/pypi/snowballstemmer
[5]http://www.nltk.org/

4.3 Stop Word Removal

After reducing all the words to their stems the words that occur very often, the so-called stop words, need to be removed because they might be identified as more significant than they truly are. Stop words are words like "the", "it", "when", etc..

The overall trend in information retrieval has gone from large stop word lists (200-300 words) to no stop words at all. A search engine with stop word removal would not be able to find anything on the band "The Who" or Shakespeare's famous "to be or not to be" as they consist solely of stop words. So before removing stop words one has to consider what the purpose is and what might be lost. In this work a list of 101 Dutch stop words from the Python package *stop_words* [6] is used. The package was used because it is available in Dutch and 17 more languages.

Additionally, two more things are done. To make sure that case-sensitivity does not get in the way all documents are converted to lowercase. Also all not-letter characters are removed, since they do not give any valuable information with this approach.

After removing stop words and bringing everything to lower case the example will look like this:

- d_1: "sun shine bright"
- d_2: "sky blue"
- d_3: "sun sky bright"
- d_4: "can see shine sun bright sun"

4.4 Term Frequency – Inverse Document Frequency (tf-idf)

Term frequency-inverse document frequency is a method to evaluate the importance of a word to a document in a corpus based on its frequency in the document and in the corpus.

We begin with calculating term frequency. The term frequency $tf(t, d)$ is simply the count how often the term t occurs in document d. This is already an interesting measure, but gives weights only based on the occurrence of a term.

The idea of tf-idf, however, is that a term is more valuable if it does not

[6]https://pypi.python.org/pypi/stop-words

only appear frequently in the current document, but it also appears only in a small number of documents throughout the corpus. This way the term adds a uniqueness to a document and it becomes a distinguishable feature of it.

For example imagine a corpus of documents about tea and the search query "green tea". The word "tea" will probably have a high frequency in every document and will thus be of no benefit since it does not allow us to exclude any documents from a search. The word "green" on the other hand is less likely appear in a document about black tea or white tea and so those documents can be excluded from the search.

This is where the inverse document frequency comes into play. The $idf(t, c)$ of the term t in the corpus c is the natural logarithm of the number of documents in the corpus divided by the number of documents that contain term t.

$$idf(t, c) = log\Big(\frac{\#documents\ in\ corpus\ c}{\#documents\ in\ corpus\ that\ contain\ t}\Big) \tag{2}$$

Since it is mostly clear which corpus is meant we will drop the corpus from the definition of the function and will always refer to the idf as $idf(t)$.

We use a smooth version of the idf to overcome a division by zero if a term is not contained in any document and a zero idf if a term is contained in every document. The smooth version bounds the idf by 1 from below through an increase of the idf by 1 and it adds an imaginary document containing every term exactly once:

$$idf_{smooth}(t) = 1 + log\Big(\frac{1 + \#documents\ in\ corpus}{1 + \#documents\ in\ corpus\ that\ contain\ t}\Big) \tag{3}$$

The tf-idf is defined as the multiplication of $tf(t, d)$ times $idf(t)$ which in our case is:

$$tfidf(t, d) = tf(t, d) \times idf_{smooth}(t) \tag{4}$$

A high weight can be achieved if the text frequency is high and the term occurs in a small number of documents.

In the example the term "sun" appears once in document d_1:

$$tf("sun", d_1) = 1$$

Overall there are four documents plus the imaginary document that contains all terms and the term "sun" appears in three of them and in the imaginary document.

$$idf("sun") = 1 + log\Big(\frac{1 + 4}{1 + 3}\Big) = 1.223$$

Thus the tf-idf of the term "sun" in document d_1 is:

$$tfidf("sun", d_1) = 1 \cdot 1.223 = 1.223$$

$$tfidf("shine", d_1) = 1 \cdot (1 + log\Big(\frac{1+4}{1+2}\Big)) = 1.511$$

$$tfidf("bright", d_1) = 1 \cdot (1 + log\Big(\frac{1+4}{1+3}\Big)) = 1.223$$

In d_1 all terms have the same frequency, but the tf-idf of "shine" is bigger, since it occurs in less documents.

The resulting vector representation with the dictionary (blue, bright, can, see, shine, sky, sun) is:

$$d_1 = (0, 1.223, 0, 0, 1.511, 0, 1.223)$$

Normalization of this vector gives:

$$\hat{d_1} = \frac{d_1}{\parallel d_1 \parallel} = \frac{d_1}{\sqrt{\sum_{i=1}^{n} d_{1i}^2}} = \frac{d_1}{\sqrt{1.496 + 2.283 + 1.496}} = \frac{d_1}{2.297}$$
$$= (0, 0.533, 0, 0, 0.658, 0, 0.533) \quad (5)$$

Translating the example documents into vectors with tf-idf weighting and normalization yields:

$$\begin{array}{c} \\ d_1 \\ d_2 \\ d_3 \\ d_4 \end{array} \begin{array}{c} \begin{array}{ccccccc} \text{blue} & \text{bright} & \text{can} & \text{see} & \text{shine} & \text{sky} & \text{sun} \end{array} \\ \begin{bmatrix} 0 & 0.533 & 0 & 0 & 0.658 & 0 & 0.533 \\ 0.785 & 0 & 0 & 0 & 0 & 0.619 & 0 \\ 0 & 0.533 & 0 & 0 & 0 & 0.658 & 0.533 \\ 0 & 0.296 & 0.463 & 0.463 & 0.365 & 0 & 0.591 \end{bmatrix} \end{array}$$

4.5 Cosine Similarity

The final step is measuring the similarity between the vector representations of the documents. The most common technique uses the cosine similarity. The idea is calculating the cosine of the angle between two vectors and the closer they are to each other, the more similar the documents are.

Comparing to other similarity measures cosine similarity performs very good as the number of dimension rises. Also since cosine similarity does not depend on the length of the vector it is not biased towards longer texts.

The similarity between two vectors $A = (a_i)_i$ and $B = (b_i)_i$ is calculated in the following way:

$$\text{similarity}(A, B) = \cos(\theta) = \frac{A \cdot B}{\parallel A \parallel \parallel B \parallel} = \frac{\sum\limits_{i=1}^{n} a_i b_i}{\sqrt{\sum\limits_{i=1}^{n} a_i^2}\sqrt{\sum\limits_{i=1}^{n} b_i^2}} \tag{6}$$

Since the vectors have already been normalized and $\parallel A \parallel = \parallel B \parallel = 1$ the calculation of the cosine similarity simplifies to:

$$\text{similarity}(A, B) = \cos(\theta) = A \cdot B = \sum_{i=1}^{n} a_i b_i \tag{7}$$

The result is a number between 0 and 1 which can be treated like a similarity percentage with 0 meaning they are not similar and 1 meaning they are the same.

Note that the model used in this work uses a bag-of-words approach, meaning that the order of the words does not influence the result. A document D_1 and its inverse document D_1^i (every term in the inverse order) will have a similarity of 1.

In the example it is now possible to calculate how similar $d1$ is to the other three documents.

For a better readability all zero times zero multiplications are ignored.

$$\begin{aligned}\text{similarity}(d_1, d_2) = 0 \cdot 0.785 + 0.533 \cdot 0 + 0.658 \cdot 0 + 0 \cdot 0.619 + 0.533 \cdot 0 \\ = 0\end{aligned} \tag{8}$$

$$\begin{aligned}&\text{similarity}(d_1, d_3) = \\ &\qquad 0.533 \cdot 0.533 + 0.658 \cdot 0 + 0 \cdot 0.658 + 0.533 \cdot 0.533 \\ &\qquad = 0.567\end{aligned} \tag{9}$$

$$\begin{aligned}&\text{similarity}(d_1, d_4) = \\ &\qquad 0.533 \cdot 0.296 + 0 \cdot 0.463 + 0 \cdot 0.463 \\ &\qquad + 0.658 \cdot 0.365 + 0.533 \cdot 0.591 \\ &\qquad = 0.713\end{aligned} \tag{10}$$

Thus it can be seen that d_4 is most similar to d_1 with a similarity score of 0.713.

For the vectorization and tf-idf calculation we use the package *scikit learn* and its module *TfidfVectorizer* [7]. For calculations of the cosine similarity we use *scikit learn*'s *pairwise_distance* module [8].

[7] http://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html

[8] http://scikit-learn.org/stable/modules/generated/sklearn.metrics.pairwise.pairwise_distances.html

5 Results

The results are obtained using the technique described above. It is possible to measure the correctness of the results because the tweets are labeled with an article id - either the article is mentioned in the tweet or the tweet is in reply to another tweet that mentions an article.

For algorithm run time reduction only articles that occurred more than 15 times are used, so our test set contains 2,903 articles and 153,634 tweets. Out of those tweets 89.27% (137,154 tweets) contain a URL. This is relevant because tweets containing a URL are often published by twitter accounts that share news or news bots and they can contain the headline of the article making them easier to identify. Correctly identifying a news article in a tweet without a URL is thus more valuable.

5.1 Similar Articles

A problem is that both nos.nl and nu.nl publish articles on the same topic. This leads to confusion when categorizing the tweets since if a tweet is originally mentioning an article written by nos.nl about the refugee crisis in Europe but the algorithm results in the tweet being more similar to the nu.nl article on the same topic, then the result is satisfying, but technically the result is counted as not correct.

To overcome this a list of similar articles is created. For this VSM with tf-idf and cosine similarity is used. Each article from nos.nl is tested against all nu.nl articles for similarity. If the nu.nl article is found to be similar with a similarity of more than 0.3 then its id is saved into a list next to the id of the nos.nl article tested. If multiple articles have a similarity of 0.3 only the article with the highest similarity is saved. This might be a limitation since, for example, in the month of June 2016 there are a lot of articles about the European Championship in football all mentioning the same players, teams and they are all more or less similar.

The result of this step is a data set with two columns - all nos.nl articles in one and the most similar nu.nl articles, if present, in the second.

The similarity score of 0.3 was chosen after running the similarity computations for part of the articles and manually checking for which similarity scores the article topic is truly equal.

Having created this data set we compare our result of the similarity algorithm not only to the real news id of the tweet, but also to its most similar news id of the other news site. This allows to successfully deal with the problem.

5.2 Time Window

Another issue is that due to the large amount of articles it is less probable that an article published in the beginning of June is discussed in the tweets from the end of June. Introducing time windows of varying sizes is used to test this hypothesis.

As discovered in section 3.2 99% of the articles do not start to appear in the tweets earlier than two hours prior to the publishing time. And almost 80% of the articles stop being discussed four days after they are published. Because we want to categorize the tweet we have to invert the time window to a tweet publishing time perspective. So the main window that is expected to perform well is between four days before a tweet is written and two hours after. We add an extra hour after to make sure all entities around the two hour mark are also considered. Three more windows are introduced for comparison. The time windows are shown in figure 3.

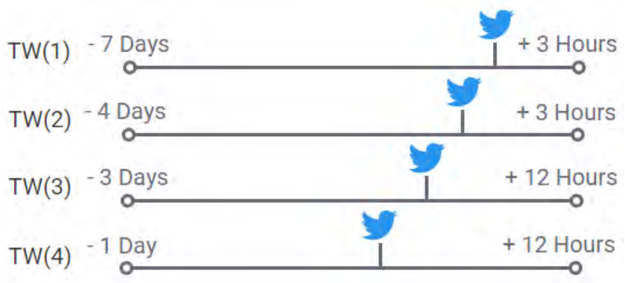

Figure 3: The four time windows used

5.3 Outcomes

Table 1 shows the percentages of tweets that are categorized correctly. An improvement can be seen through using time windows with the window of minus four days to plus three hours being the best overall. As expected the results on tweets that include the URL are better. For the tweets not containing the URL surprisingly the smallest window of minus one day to plus twelve hours outperforms the others. Although overall the performance on tweets not containing the URL is poor.

Table 1: Percentage of correctly by tf-idf categorized tweets

	no window	-7d/+3h	-4d/+3h	-3d/+12h	-1d/+12h
all tweets	60.56%	68.66%	69.63%	68.13%	67.18%
with URL	66.78%	75.17%	76.01%	74.32%	72.96%
without URL	8.77%	14.48%	16.58%	16.61%	19.12%

Until now the similarity score has not been taken into account.

Looking closer at the similarity score shows in Figure 4 that the correctly categorized tweets have mostly a similarity score above 0.2 while the wrongly categorized tweets have a similarity score below 0.2. We conclude that if the similarity score is below 0.2 the evidence of the tweet belonging to an article is too poor.

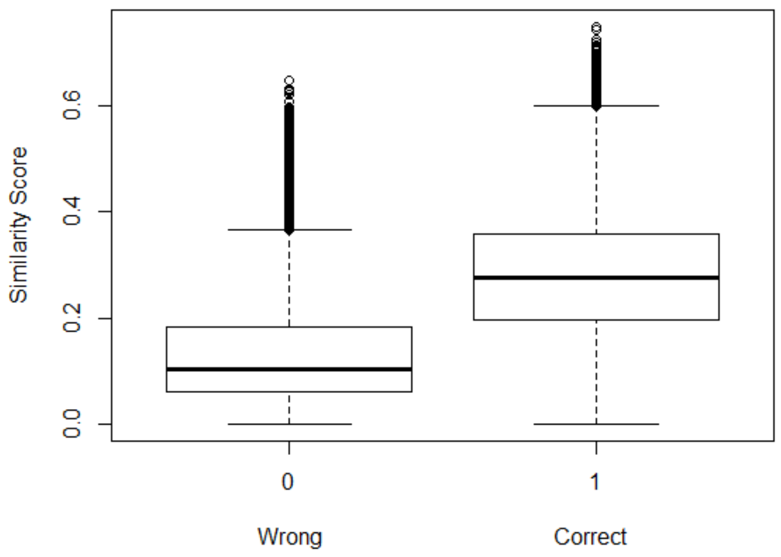

Figure 4: Similarity Score of wrongly/correctly categorized tweets

Table 2 shows the results for tweets where the similarity score is above 0.2. With the time window of minus four days to plus three hours it is possible to categorize over 90% of the tweets correctly. The same holds for the tweets containing the URL. Also the results for tweets without a URL jump up remarkably to over 57% for the time window of four hours (TW(1)). For tweets not containing the URL the time window of four days is outperformed by the time window of one day (TW(4)).

Table 2: Percentage of correctly by tf-idf categorized tweets whith a similarity score above 0.2

	no window	-7d/+3h	-4d/+3h	-3d/+12h	-1d/+12h
all tweets	79.96%	89.44%	90.43%	88.75%	89.87%
with URL	81.03%	89.95%	90.82%	89.11%	90.10%
without URL	29.34%	50.75%	57.65%	56.88%	67.82%

A similarity score above 0.2 is achieved for 56.57% of the tweets.

6 Conclusions

This research paper aimed to identify the tweets belonging to a given news article.

A vector space model combined with tf-idf weighting and a cosine similarity was used for this purpose. This approach resulted in an accuracy of 80% overall and 30% for tweets that did not include the URL of the news article.

Time windows were introduced to test the hypothesis that including articles whose publishing date is far away from the tweet date might trigger false categorizations. The time window of 4 days was found to be most effective with an overall accuracy of 90.82% and an accuracy for tweets without the URL of 57.65%. Surprisingly for tweets not containing the URL the most accurate time window was the one of one day with 67.82%. This can be explained because 90% of tweets not containing a URL are published less than a day after the article. Overall time windows always outperformed the results without the time window proving their use to be effective.

Comparing to the work of Vikre and Wold [12] the precision achieved in their work is 91.6% and thus much higher. Also they do not have a lot of tweets that they can not categorize. Their approach is more general while this approach is strongly tied to the news articles that were extracted.

Possible limitations of this work are the lacking possibility of this approach to deal with synonyms. The words "disaster" and "catastrophe" have zero similarity with this approach despite having the same meaning.

Another possible limitation is the bag of words approach. The order of the words in the sentence has no influence on the similarity, thus the two sentences "Mike is faster than Molly" and "Molly is faster than Mike" are regarded equal although their meaning is completely the opposite.

Additionally tweet authors do not always stick to the rules of grammar which might pose a limitation on the success of the stemming algorithm. If the stemming algorithm is not able to recognize the word and stem it correctly then word frequencies will be false and thus results will be distorted.

The examination of the effect of these limitations brings an opportunity for the extension of this work.

Furthermore by using the results of this work it is possible to find more tweets related to known news topics. With this extended amount of data the initial goal of studying the spread of news in the dutch tweetosphere can be achieved.

References

[1] Bandari, R., Asur, S., and Huberman, B. A. The pulse of news in social media: Forecasting popularity. In *In ICWSM* (2012).

[2] Becker, H., Naaman, M., and Gravano, L. Beyond trending topics: Real-world event identification on twitter. In *Fifth International AAAI Conference on Weblogs and Social Media* (2011).

[3] Bhattacharya, D., and Ram, S. Sharing news articles using 140 characters: A diffusion analysis on twitter. In *ASONAM* (2012), IEEE Computer Society, pp. 966–971.

[4] Chew, C., and Eysenbach, G. Eysenbach g: Pandemics in the age of twitter: content analysis of tweets during the 2009 h1n1 outbreak. *PloS One* (2010), 5–11.

[5] Hansen, L. K., Arvidsson, A., Nielsen, F. r., Colleoni, E., and Etter, M. Good friends, bad news - affect and virality in twitter. *CoRR* (2011).

[6] Hughes, A. L., and Palen, L. Hughes et al. twitter adoption and use in crisis twitter adoption and use in mass convergence and emergency events, 2009.

[7] Kunneman, F., and van den Bosch, A. Event detection in twitter: A machine-learning approach based on term pivoting. In *Proceedings of the 26th Benelux Conference on Artificial Intelligence* (2014).

[8] Kwak, H., Lee, C., Park, H., and Moon, S. What is twitter, a social network or a news media?, 2010.

[9] Rosenstiel, T., Loker, J. S. K., Ivancin, M., and Kjarval, N. Twitter and the news: How people use the social network to learn about the world.

[10] Sakaki, T., Okazaki, M., and Matsuo, Y. Earthquake shakes twitter users: Real-time event detection by social sensors. In *In Proceedings of the Nineteenth International WWW Conference (WWW2010). ACM* (2010).

[11] ten Thij, M., Bhulai, S., and Kampstra, P. Circadian patterns in twitter.

[12] Vikre, L. C., and Wold, H. M. Online news detection on twitter. Master's thesis, Norwegian University of Science and Technology, 2015.