DEPARTMENT OF INFORMATICS
PROGRAMME OF POSTGRADUATE STUDIES
IN COMPUTER SCIENCE

M.Sc. Thesis

"A Personalised System to Predict Retweets"

Michail Vougioukas
EY1417

Supervisors: Ion Androutsopoulos, Georgios Paliouras

Athens, February 2016

Acknowledgements

I would like to thank my supervisor, Ion Androutsopoulos, for his restless guidance, genuine interest, substantial help and the overall support throughout this thesis, as well as during the senior years of my studies.
I would also like to thank my second supervisor, Georgios Paliouras, for the precious advice, constructive comments and time he devoted.
Moreover, I would like to thank the Natural Language Processing Group of AUEB's Department of Informatics for all their ideas, discussions and help.
A big and special thanks goes to my family, especially Katerina, who stood by me and made everything possible over the years.
Last, but not least, I thank my friends for the patience, understanding and inspiration they offered me, during my studies.

Abstract

Information on social media is ever increasing, but much of the information provided is often of very little, if any, importance to users. In this thesis we tried to develop a system of personalised tweet recommendation, by using a mechanism to predict how likely it is for a specific tweet to be retweeted by a specific user. Previous work has shown that retweeting may be a soft signal that a tweet was interesting to a user, which can be used to train recommendation systems. Furthermore, our system, although personalised, was designed to be trainable on data aggregated over multiple users, offering a single, global model, which is nevertheless used for personalised, rather than global, predictions of retweets. We present a full study of a retweet likelihood classifier, including dataset aggregation, feature engineering and feature/model selection.

Contents

1. Introduction

1.1 Goal of the thesis

A typical Twitter user is shown hundreds or thousands of new tweets daily on their Home Timeline. While the number of tweets shown varies with the number and types of friends, location, time and active trending topics in the network, users often receive much more information than they are able to consume. This is even more intense for users who follow many and/or very active accounts (e.g. news portal accounts). As a consequence, bits of interesting information may be ignored or overlooked by a time-constrained user, who is overwhelmed by irrelevant information. Meanwhile, Twitter does not seem to currently provide large-scale, personalised content filtering in the way other networks, such as Facebook, do, in order to try to address this problem.

The goal of this thesis is to contribute towards personalised content filters for Twitter. In this context, we developed a system that classifies a tweet as *interesting* or *not interesting*, as perceived by a specific user, in a rather indirect way;by predicting whether the user will *retweet* it or not. This approach makes it possible to train machine learning algorithms using large numbers of tweets that were retweeted or not in the past. This contrasts to more direct approaches, which need a user-provided direct signal on whether a tweet was interesting or not (e.g. labelling each tweet with an interest score on a 1-5 scale), in which case obtaining training data becomes much more difficult [Vougioukas, 2014], [Alonso, Marshall, Najork, 2013], [Meier, Elsweiler, Wilson, 2014]. Our approach is based on the idea that if a user retweets a tweet, then they found it interesting [Uysal, Bruce Croft, 2011]. The reverse does not necessarily hold, because on numerous occasions there is no guarantee that a non-retweeted tweet was actually ever seen by the user. Nevertheless, one hopes that the retweeted tweets provide enough information to learn the interests of each user.

In trying to achieve our goal, we also propose the use of a single, global model, trained over multiple users' data. Personalisation in this approach is possible by introducing features modelling the user-receiver. By using a single model we aspire to capture many of the patterns existing in the entire population and to offer filters performing well enough from day one, without first requiring any manual annotation from each user.

The system can then be used both to (indirectly) identify interesting tweets for a user, as well as to predict “retweetability” (likelihood that a user will retweet a tweet), which is a common use case on its own. Clearly, by identifying interesting content, it would be possible to decrease the incoming information stream rates of users, by filtering out irrelevant tweets, and consequently to prevent interesting information from being ignored.

1.2 Structure of the remainder of the thesis

This thesis is organised as follows:

- Chapter 2 presents previous related work.
- Chapter 3 describes the proposed system, its architecture, features and tools that were used.
- Chapter 4 describes the datasets we used, the experiments we performed in order to choose the best possible version of our system, along with the conclusions of our experiments. It also provides a description of the evaluation framework and the evaluation measures we used.
- Chapter 5 summarises our conclusions and proposes possible future work.

2. Background and related work

2.1 Overview

There have been two main approaches to detecting interesting content in microblogs, such as Twitter: global filters and personalised filters. The former try to predict tweet interestingness for the entire social network or, at least, a broad audience, while the latter aim to provide recommendations which are solely relevant to a specific user's own interests and preferences. While the two approaches do not share identical sets of motivations and use cases, they are clearly related because, from an abstract point of view, they address the same problem: how to reduce the volume of uninteresting content in microblog streams. Furthermore, studying either approach gives useful input for the other. In the case of this thesis, some inherent shortcomings of global filters (reported below) motivated us to prefer developing personalised filters. Retweets serve as an indirect way of signalling interesting content and can be used in both the aforementioned filtering approaches, as an alternative to requiring users to provide explicit interest scores for past tweets.

2.2 Global filters

Global filters [Alonso, Marshall, Najork, 2013] aim to identify content which is generally interesting for a large set of users, without identifying preferences of specific users.

Hurlock and Wilson [Hurlock, Wilson, 2011] present a qualitative investigation of factors believed to affect the perceived interest of a tweet. They offer useful prediction features (e.g. existence of URL, tweet length), which are also relevant for the task of personalised tweet filtering. Duan et al. [Duan, Jiang, Qin, Zhou, Shum, 2010] propose a method to move from the temporal ordering of tweets, as employed in Twitter today, to a relevance-and-authority-based ranking.

Conclusions that can be drawn from work in the area include that global filters may have useful applications in big data storage planning (for instance, a high-relevance-first priority in storage and replication), as well as in providing better results in microblog search engines. On the other hand, most studies conclude that there is no universal agreement on what is interesting and what is not. Even when all other factors (annotation and label quality) are optimised, global filters achieve very little agreement with humans. Interestingly enough, even humans achieve very little inter-rater agreement themselves and this does not seem to improve when using more interesting tweet sets (e.g. news), instead of random tweet sets. In brief, users may agree that a certain category contains, on average, more interesting tweets, but they do not agree on what is interesting.

For these reasons, we believe that predicting the interest of tweets from a global viewpoint is a difficult task. Consequently, this thesis opted to work on personalised filters, where no agreement between humans is required. Although we utilised ideas (e.g. features) from studies on global filters.

2.3 Personal filters

A personal (or personalised) filter has the goal to predict interesting content, taking into account a single user's personal preferences. Personal filters can take the form of pass/fail filters, ranking mechanisms or recommender systems; the latter focus on positive example discovery, where high recall is not crucial.

In previous work [Vougioukas, 2014], we worked on a method to develop personal filters, using tweets from timelines of six users annotated with interest scores by the users themselves. Each filter was trained and tested on tweets from the timeline of a particular user. All the filters were of the same type and they used the same features, but they were independent of each other otherwise. Annotation turned out to be a bottleneck, because users had very little motivation to label many tweets with interest scores in a short period of time. We could not obtain more than 1000 labelled tweets per user and it took almost two months to complete the annotation, despite using a user-friendly annotation interface. As a result of separate filters and annotation difficulties, we were unable to make full use of the data we collected and it was impossible to evaluate the system with a big number of test users. Thus, we believe that our previous approach would not scale well in real conditions. Moreover, it does not address the *cold start* problem, where a filter must be provided to a new user, with no training data available for this user.

Uysal and Croft [Uysal, Croft, 2011] also aimed to devise effective filtering mechanisms, in the form of personalised tweet ranking. Unlike our own previous work, they used previous retweets to learn to place more important tweets in higher list positions. As a side use case, they also use their method to rank users based on their likelihood of retweeting tweets, which may be a useful extension for content authors, not receivers. Uysal and Croft also studied the correlation between retweeting and the actual level of perceived interest. Even though the sample of their pilot study was too small to make sound conclusions, preliminary results seemed to indicate a significant correlation. Interestingly, very different F1 scores were achieved with different feature groups, with content-based features achieving 0.04, and author-based ones 0.3. Chen et al. [Chen, Nairn, Nelson, Bernstein, Chi, 2010] proposed a tweet-based recommender system, focusing on detecting interesting URLs in tweets, in real time. They defined three dimensions of the problem: how to select candidate URLs from a tweet stream to recommend, how to use content information, in terms of relevance with the users' topics and how to use social information, such as how popular a tweet containing a URL is in a neighbourhood of a user's followees-of-followees. The multiple approaches

followed to address each one of these dimensions resulted in twelve different combinations that can be used as the recommender system's engine. This work offers algorithms and ideas general enough to be used in a plethora of other information streams, as they assume little that is specific to Twitter.

Overall, previous work in personalised filters gives us many ideas of features to use. There is a wide agreement on certain basic factors affecting a tweet's level of interestingness, although the list of factors is by no means exhaustive. These factors usually include the existence of a URL in the tweet (a URL often makes a tweet more informative), tweet length (users tend to prefer tweets with more information) and source authority (influential users tend to write tweets which become more popular). The recency of a tweet is also a very essential factor. Moreover, the previous interaction (if any) between the author and the receiver of the tweet often plays an important role. More generally, user modelling and user similarity measuring are also important. Finally, users who follow many users tend to need stricter filtering than users who follow fewer users.

Following Uysal and Croft [2011], we aim to predict which tweets will be retweeted, using a global filter trained on data from multiple users. The features that we use, however, are user-sensitive, which allows the decisions of a global filter to be personalised, i.e., our global filter may predict that the same tweet will be retweeted by one user but not another. Our approach combines personalised filtering with the promising properties of systems that predict retweets and the quality of models trained on large amounts of data from a large number of users. Furthermore, our approach allows for system evaluation over a much larger number of test users, compared to our previous work.

3. System and tools

3.1 The central idea

For reasons discussed earlier and also presented in our previous work [Vougioukas, 2014], our motivation for social media personalisation persisted and, as a result, the current work also tries to develop *personalised filters for Twitter*. Nevertheless, we have experienced that the work needed to adequately train such a system with “traditional” means (separate model and training corpus for each user, annotation by humans) is very challenging, due to difficulty of persuading users to spend time annotating tweets with the interest scores. At the same time, we wanted to exploit large datasets of tweets from multiple users that do not contain, however, interest scores directly.

Towards the direction shaped above, we first re-examined what exactly we intended to predict. In previous work, we modelled a tweet's relevance to a user's interest with an one-to-five (later, one-to-three) scale. Although multiclass prediction is a more difficult task than binary prediction, we managed to achieve accuracy well over a baseline system. On the other hand, we realised that we had collected a lot of tweets for our development users, but they only annotated a fraction with interest scores, which could be used for supervised learning. In this thesis, we do not try to predict interest scores, but whether or not a tweet will be retweeted by a particular user, using a binary classifier.

As already noted, Uysal and Croft [2011] were among the first to study the retweet behaviour of users in the context of personalised tweet ranking. The authors argue that retweeting may indicate that the user found a tweet interesting. While not retweeting may not always indicate that they did not find the tweet interesting (e.g. not seeing a tweet is an independent, valid reason for not retweeting it), certain sound assumptions can be made, allowing us to fuzzily detect under which conditions such “false alarms” may appear. The same assumptions make it possible to provide negative examples to the model trainer. More specifically, we can assume that if a user has retweeted a friend's tweet at least once before, then the user regularly sees this friend's tweets. Hence, not retweeting one of this friend's tweets can safely be considered as a conscious action. We also discard tweets from very inactive friends, whose tweets are very unlikely to be read. As a result, we managed to overcome the annotation bottleneck, since now labels (retweeted or not by the particular user-recipient) could automatically be induced from data, which allowed us to build much larger annotated datasets for each user.

After the prediction target switch, we examined how we could profit from large, multi-user datasets typically used in global filter development. We decided to build a *single* prediction model for all users, rather than *specialised* models, different per user. Although it may seem like a global filtering approach, our

method can produce personalised (different per user-recipient) decisions, because we also use user-aware features.

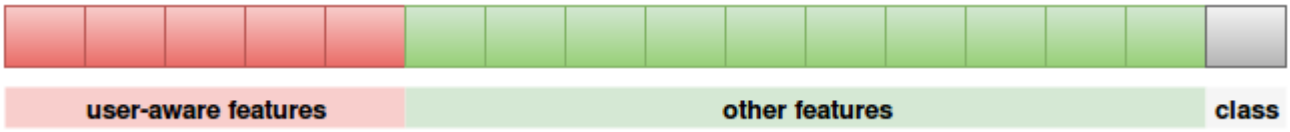

Figure 1: A feature vector representing a tweet received by a particular user

Figure 1 sketches a feature vector representing a tweet received by a particular user. Some features are user-aware, while others are not. User-aware feature values depend on the user who uses the single, global model. Therefore, the same tweet has different feature vector representations for different users. Depending on the quality of the user-aware features used, we believe that they can play a significant role in determining the class value. This would result in predictions that are, by definition, personalised.

Another important factor affecting the outcome of this approach is the training data quality and diversity. Intuitively, training data aggregated over a large number of users, are more likely to reveal a bigger and more realistic picture of the population patterns. Nevertheless, our study does not claim to provide a system working under any circumstances, for any type of user, in any topic. In contrast, we developed our approach in a more controlled way. We first tried to work using data with very few topics (SNOW dataset) and our final implementation presented in this report, uses a dataset where all users are journalists (*terms "user" and "journalist" will hence be used interchangeably*). As a consequence, some of our conclusions may not be directly applicable to domains with different user types, although they can still provide useful input in similar systems developed for other contexts.

Concerning our system's use cases, its basic one is the task of filtering tweets, by discriminating instances between the "will be retweeted" and "will not be retweeted" classes, using a fixed, though possible to tune, threshold (pass/fail decision). In an application, tweets placed in the negative class, could be demoted to less viewable user interface positions, or they could be hidden. Inspired by the work of Waldner and Vassileva [Waldner, Vassileva, 2014], we decided to also study the use case of tweet ranking, using the classifier confidence for the positive class to rank tweets. This is a more flexible system output, as it allows user access to as many tweets as desired, with the most important ones always on top. A third interesting use case is predicting retweet likelihood, though outside the personalised filtering context we have so far discussed. Predicting retweetability alone, could be useful for influential content creators, journalists, politicians, companies, online marketing and generally whenever a piece of content is desired to obtain significant propagation and impact.

3.2 System architecture

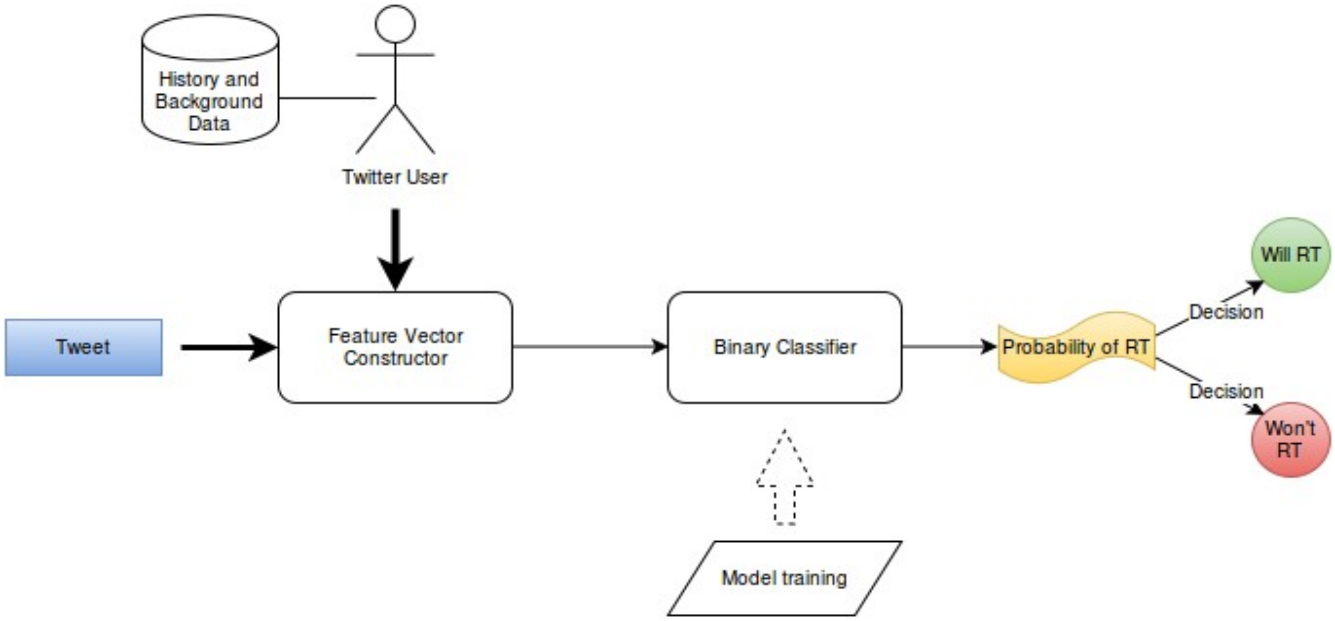

Figure 2: The system architecture of this thesis

The system of this thesis predicts how likely it is that a given user will retweet a given tweet. Hence, its inputs are a tweet and a Twitter user, while its output is either a probability of retweeting, or a hard decision based on the probability, according to the use case.

From a programming point of view, a tweet object comprises its main *text* field and other metadata, loosely following the Twitter API object definition. Similarly, a Twitter user object comprises various metadata, as in Twitter API, but is also linked with an information base containing user's tweet collections, network relationships, network influence data, history of interactions etc.

The information above is combined with global information (e.g. keyword lists, vocabularies) and preprocessors (e.g. text normaliser) by the Feature Vector Constructor, where an input tweet is transformed into a vector suitable for consumption by the classifier in the subsequent stage. Developing of the Feature Vector Constructor mainly involved feature engineering, feature selection and the implementation in Java.

The vector from the stage above is then passed to the binary classifier, which is the prediction engine of the system. The classifier includes the trained binary prediction model (e.g. logistic regression, decision tree, decision forest) and produces a probability distribution over the two classes.

Depending on the use case, the system finally produces either a list of tweets ranked by decreasing likelihood that they will be retweeted, or two non-

overlapping sets of tweets, corresponding to the two classes. For the rank-based use case, tweets are ranked by decreasing classifier confidence of the positive class, while for the set-based one, tweets are split by the classifier confidence of the positive class, using a threshold (default: 0.5).

The classifier's model is trained on datasets aggregated over many users (but, as explained, the model produces personalised decisions), while the learning algorithm varies from one model to another. Training can be executed once or be repeated periodically, taking into account new data instances, or using a sliding time window on data.

Developing of the binary classifier mainly involved model selection and validation using different volumes of training data, as well as training data with various ratios of positive class examples. The model and learning algorithm implementations were imported from the Weka[1] software [Hall, Frank, Holmes, Pfahringer, Reutemann, Witten, 2009].

3.3 Data representation and features

Since the system's central component is a standard Machine Learning classifier, all data instances (tweets) need to be represented by *feature vectors*. This includes training, development and testing instances (with known class labels), as well as new instances submitted for classification to a production filter (with unknown class labels).

Each feature vector contains up to 50 feature values and a binary class label, {Retweet, No_Retweet}, which is the observation/prediction of whether the user retweeted/will retweet a tweet under examination, or not. Each of the 50 candidate features we propose is considered a factor that possibly affects a tweet's retweetability (as perceived by a user's viewpoint) and the features were engineered by studying what the Twitter API offers, studying previous work and taking advantage of our previous experience. To better understand the problem space (factors, actors, objects), the features can be grouped as in the visualisation below. This grouping can also be useful in engineering new features, as well as in evaluating the features on a group basis, by holding out one or more specific groups during development tests.

1 http://www.cs.waikato.ac.nz/ml/weka/

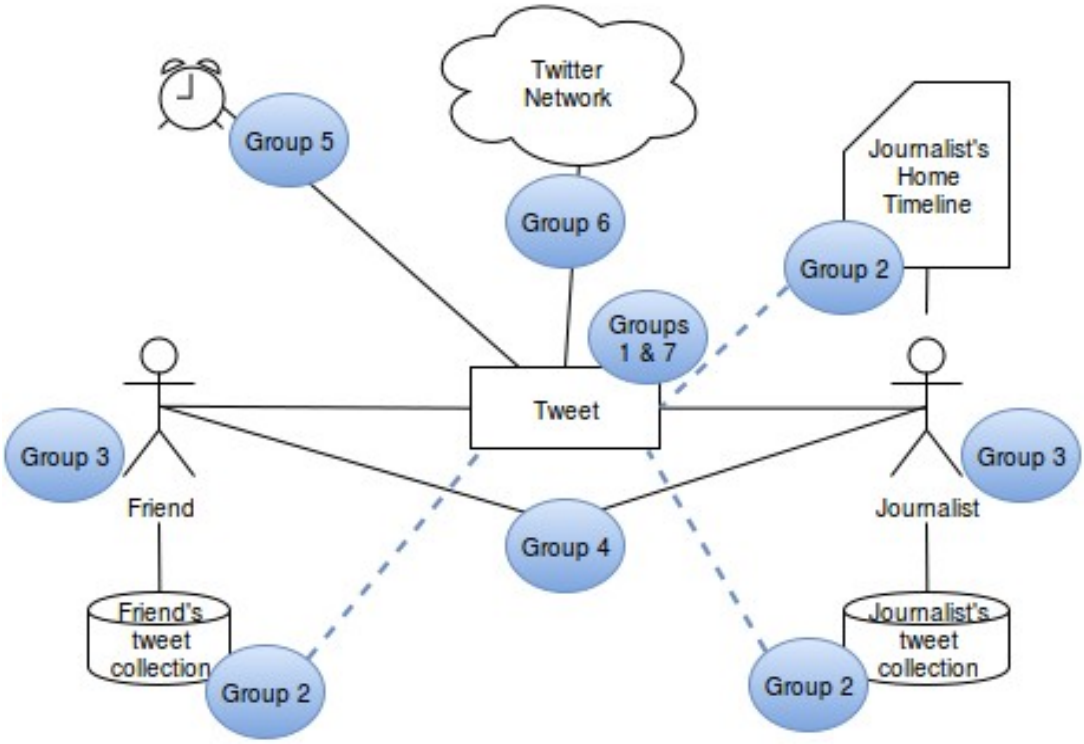

Figure 3: Visualisation of the problem space, feature groups and relations

In Fig. 3, a tweet is primarily associated with a journalist who receives it and a friend who writes or simply propagates it. In the latter case, the original tweet is written by a third party and the friend of journalist retweets it. In all cases, we call the friend of the journalist the *source* of the tweet, whether this friend is the original tweet author, or not. Furthermore, we define as *posting* an action meaning either writing an original tweet, or simply propagating, by retweeting, a tweet written by a third party. With these in mind, there are seven feature groups, as follows:

- In *Group 1* there are features which describe properties of the tweet/instance itself. Group 1 includes the features:
    - *Tweet length in characters [FT1]*
    - *Does the tweet contain a URL? [FT2]*
    - *Does the tweet contain a mention? [FT3]*
    - *Does the tweet contain a hashtag? [FT4]*
    - *Global retweet count [FT5]*
    - *Global favourite count [FT6]*
    - *Does the tweet contain an exclamation mark? [FT7]*
    - *Does the tweet contain a photo? [FT8]*
    - *Number of mentions in the tweet [FT9]*

- In *Group 2* there are features which are cosine similarity measurements between the tweet/instance and certain important tweet collections. These features are based on the idea that tweets that are very different (novelty) or very similar (resemblance) to collections of older tweets, are often very desirable or very undesirable for a user, therefore these similarities are highly correlated with the class label. Similarities are calculated using the TFIDF Bag-of-Words representation of each tweet's (normalised) text. Furthermore, a single tweet's similarity with a tweet collection is defined as the average similarity of the tweet with all the tweets in the collection. Group 2 includes the features:

    - *Similarity between the tweet and all other tweets (both original and retweeted ones) posted by the source friend [FT10]*
    - *Similarity between the tweet and all tweets posted by the journalist [FT11]*
    - *Similarity between the tweet and all tweets received by the journalist on their Home Timeline [FT12]*
    - *Similarity between the tweet and tweets previously retweeted by the journalist [FT13]*

- In *Group 3* there are features modelling the network influence, popularity and authority of the friend and the receiving journalist. These features include standard user account statistics, as recorded by Twitter, as well as measurements obtained from the Klout[2] social analytics service. Group 3 includes the features:

    - *Number of followers of the friend (users following the friend) [FT14]*
    - *Number of friends of the friend (users followed by the friend) [FT15]*
    - *Number of tweets of the friend [FT16]*
    - *Number of lists of the friend [FT17]*
    - *Is the friend a verified account? [FT18]*
    - *Number of days the friend's account has been active for [FT19]*
    - *Does the friend have a URL in their user description? [FT20]*
    - *Klout score (influence) of the friend [FT21]*
    - *Delta of the Klout score of the friend, over last day [FT22]*
    - *Delta of the Klout score of the friend, over last week [FT23]*
    - *Delta of the Klout score of the friend, over last month [FT24]*
    - *Number of followers of the journalist [FT25]*
    - *Number of friends of the journalist [FT26]*
    - *Number of tweets of the journalist [FT27]*
    - *Number of lists of the journalist [FT28]*
    - *Is the journalist a verified account? [FT29]*

2 http://klout.com/ :Klout is a service that estimates a user's social influence, by taking into account their activity in various social networks.

- *Number of days the journalist's account has been active for [FT30]*
- *Does the journalist have a URL in their user description? [FT31]*
- *Klout score (influence) of the journalist [FT32]*
- *Delta of the Klout score of the journalist, over last day [FT33]*
- *Delta of the Klout score of the journalist, over last week [FT34]*
- *Delta of the Klout score of the journalist, over last month [FT35]*

- In *Group 4* there are features relevant to previous recorded interaction between the friend and the journalist, in addition to the interaction event generated by the tweet/instance being considered. Group 4 includes the features:

  - *Is the journalist mentioned in the tweet? [FT36]*
  - *Has the friend ever mentioned the journalist before? [FT37]*
  - *Has the journalist ever mentioned the friend before? [FT38]*
  - *Has the friend ever retweeted a tweet posted by the journalist before? [FT39]*
  - *Has the journalist ever retweeted a tweet posted by the friend before? [FT40]*
  - *Number of times the journalist has previously retweeted tweets posted by the friend [FT41]*

- In *Group 5* there are features which take into account the timing of the tweet. A tweet that is very similar (or identical) to other recently received tweets may be “old news”. Group 5 includes the features:

  - *Similarity between the tweet and tweets received by the journalist during the week before [FT42]*
  - *Similarity between the tweet and tweets previously retweeted by the journalist during the week before [FT43]*

- In *Group 6* there are features relevant to the association of the tweet with the journalist's immediate network (one-hop neighbours). Group 6 includes the features:

  - *Is the original tweet author a friend of the journalist? [FT44]*
  - *Number of times the tweet has been retweeted by friends of the journalist [FT45]*

- In *Group 7* there are features modelling the tweet's wording and phrasing. According to Tan et al. [Tan, Lee, Pang, 2014], the way a tweet is worded, rather than its actual information content, may have a significant impact on message propagation. Interestingly enough, Tan et al. found out that in pairs of tweets containing exactly the same information (e.g. a URL), one of the two tweets was propagated more intensively than the other, because of

their different wordings. Factors believed to affect wording quality include the use of specific keywords, parts of speech, etc. Like Group 1 features, these features also depend only on the specific tweet being considered itself. Group 7 includes the features:

- *Number of keywords explicitly asking to retweet or share the tweet (e.g. “RT”, “please spread”, “share”) [FT46]*
- *Number of nouns and verbs in the tweet [FT47]*
- *Number of definite articles in the tweet [FT48]*
- *Number of indefinite articles in the tweet [FT49]*
- *Score of keyword list. Tan et al. offer a list of 20 specific “good” keywords which are believed to increase a message's propagation probability and a list of 20 “bad” keywords believed to decrease it. In our score, each good keyword's existence adds 1, while each bad keyword's existence subtracts 1. If none of the list keywords exist, this score is 0. [FT50]*

By employing feature selection (see next chapter), it is possible to keep only the best out of the fifty candidate features, e.g. only the feature subset with the lowest inter-feature correlation and the highest correlation with the class label, or the top features, based on Information Gain evaluation. This could lead to system speed-up, as well as to better generalisation ability, since having fewer features usually reduces overfitting on the training data.

3.4 Data preprocessing and other tools

3.4.1 Preprocessing

The *text* field of each instance/tweet (the tweet's text, including mentions, hashtags and “RT” token, if one) is normalised before use. For instance, two tweets which only differ in a URL (e.g. because their authors use different URL shorteners) should generally be considered identical. On the other hand, care must be taken in order not to eliminate important differences during the preprocessing. For example, a happy and a sad smiley cannot be mapped to a general object category *smiley*. Such a normalization is also important when using word embeddings [Mikolov, Chen, Corrado, Dean, 2013], [Mikolov, Sutskever, Chen, Corrado, Dean, 2013], [Pennington, Socher, Manning, 2014]. For example, there may be a single generic embedding for all URLs. For the aforementioned reasons, each tweet's text is normalised by our system, following the preprocessing steps below, which are based on the preprocessor used when producing GloVe[3] embeddings:

3 http://nlp.stanford.edu/projects/glove/

- Regular-expression-based (RegEx-based) replacement of all URLs with a generic string, “<URL>”
- RegEx-based replacement of all numbers with a generic string, “<NUMBER>”
- RegEx-based replacement of all smileys with the suitable generic smiley string, according to the following sentiment-based grouping:
  - “<HEART>”, for love/like-meaning smileys (e.g. “<3”)
  - “<SMILE>”, for positive sentiment smileys (e.g. “:-)”)
  - “<SADFACE>”, for negative sentiment smileys (e.g. “:-(”)
  - “<NEUTRALFACE>”, for neutral sentiment smileys (e.g. “:-|”)
- Conversion of text to lower case

Notes:
→Because of the conversion to lower case, no named-entity recogniser can be used after the steps above have been applied.
→Mentions and hashtags should also be converted to the generic “<MENTION>” and “<HASHTAG>” tokens respectively, if word embeddings are used for text representation. When using the Bag-of-Words model we believe a mention or a hashtag bears important semantics, unless it is a stopword.
→Whenever tokenization is needed (e.g. when converting tweet tokens to word embeddings or when counting frequency of specific keywords), the CMU ARK Twokenize[4] tokenizer is applied.
→For POS-tagging, the CMU ARK POS-tagger is applied.
→We actually built an English-only system, by filtering out non-English tweets. The system could, in theory, be extended to any other language, if suitable tools and embeddings are available.

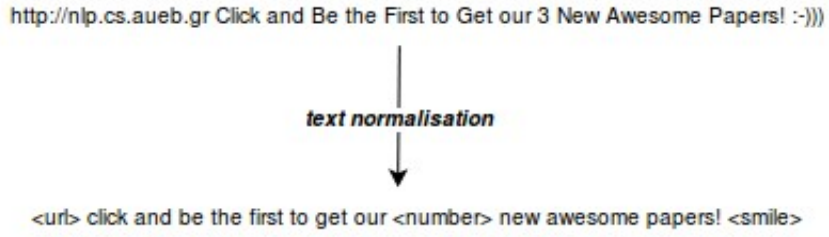

Figure 4: Example of text normalisation

3.4.2 Using Apache Lucene

Our system's main implementation uses a TFIDF Bag-of-Words representation for tweet texts. In order to index the tweet texts and later retrieve their term vectors, our system employs the Apache Lucene 5.4 search engine. In this case, the tokenization step of preprocessing is not applied and is undertaken internally by Lucene, along with other preprocessing tasks, using the StandardAnalyzer of

4 http://www.cs.cmu.edu/~ark/TweetNLP/

Lucene.

By using Lucene, our system is able to retrieve tweet texts and calculate cosine similarities efficiently. In order to minimize the time needed for a similarity estimation, we take advantage of the TFIDF vector sparsity. We exploit the fact that only TFIDF scores of the *common* terms of the two instances are needed in calculating the dot product. In addition to these, we also need TFIDF scores of terms that exist in only one of the two instances in order to calculate each vector's norm. As a consequence, on each similarity estimation we only need to calculate very few TFIDF scores and do operations on small, usually dense TFIDF vectors.

4. Data and experiments

4.1 Data

As explained in Chapter 3, our system is trained and tested using datasets aggregated over multiple users. Nevertheless, we decided to focus on certain topics or user types, rather than trying to develop an over-optimistically general system. The data we collected follow, therefore, this consideration. Annotation is not an issue in our case, because as already explained, the target labels (retweeted or not retweeted) are available.

Any dataset considered in the context of this study should generally follow these *principles*:

1. It should contain tweets for a large number of users (at least in the order of hundreds), so that the system can be trained and tested on a reasonably representative data sample.
2. For each user, it should contain at least 500 positive examples (retweeted tweets) and a comparable number of negatives, so that acceptable performance can be reached. The minimum requirement of examples is set to 500, following our previous experience [Vougioukas, 2014].
3. It should contain only tweets in English. In our implementation, this is checked using the *lang* field provided by the Twitter API, but language identification methods (e.g. based on language models) could have been applied instead.

4.1.1 The SNOW dataset

The SNOW dataset is a corpus of around 1.1 million tweets, authored by 580 thousand different users, collected by submitting the queries “syria”, “terror”, “ukraine”, “bitcoin” to Twitter Search. The corpus was constructed in the context of the SocialSensor[5] project, for a different task. Around 60% of the tweets are retweets. The dataset is accompanied by a list of influence scores for the 580 thousand authors, which can also be used in relevant features.

A motivation for using this dataset was the existence of the four topic-like subgroups. However, the dataset did not offer a mapping between the four query terms and the tweets returned by each query. In order to avoid using unsupervised topic detection methods in such an early stage, we opted to allocate the tweets in four groups (“topics”), based of which query term they explicitly contained. Due to latent results returned by Twitter Search (results which were found to be relevant, but without explicitly containing the query term), this turned out to be a bad approach, as only 30% of tweets contained one of the four query

5 http://www.socialsensor.eu/

terms. As a result of our topic allocation method's failure to handle latent results, we had to discard 70% of the dataset.

Subsequently, the language filtering removed a further 8% of the initial tweets. In this filtering we only kept tweets marked as English, but also tweets with undefined language, because they were very few. We also observed that tweets of undefined language had almost always zero probability of retweet, which could make them useful negative examples for the classifier. As a result of the aforementioned processes, the dataset was reduced as follows:

Total tweets	**253,875**
Topic “syria”	48,806
Topic “terror”	15,694
Topic “ukraine”	114,623
Topic “bitcoin”	74,752

Table 1: Number of tweets per topic in the SNOW dataset after filtering

While the topic sizes are imbalanced, they were initially considered adequate. Later, we observed that our dataset users are numerous, but there are very few tweets per user and, hence, the second dataset principle is not satisfied. Specifically, almost 98% of users had at most 10 tweets in the corpus. Given time constraints, there was no realistic chance to extend the dataset by importing new tweets from the Twitter API, especially for such a large number of different users.

In conclusion, we decided not to use the SNOW dataset for our task. We tend to believe that datasets for personalised systems should be aggregated in a more user-centric way than SNOW, which is a purely topic-centric dataset. It should be noted, however, that SNOW has already been used in other tasks.

4.1.2 The final Twitter dataset

Moving on from SNOW, we decided to use a dataset from the work of Zamani et al. [Zamani, Paliouras, Vogiatzis, 2015], previously used to identify users on various social networks. This dataset contained data for 262 well-known journalists (thus, all development users belong to a specific type of user) and comprised Twitter profile information, as well as the 3,200 most recent tweets authored or retweeted by each journalist.

While the dataset above offers a pool of positive examples for each user (retweets), we also needed to download tweets authored or retweeted by friends of each user, out of which to extract a number of negative examples. To avoid a time shift between positive and negative examples (as the initial dataset had been aggregated a while before), we decided to also download again the updated

profile information and 3,200 most recent tweets, for each journalist. As a consequence, our final dataset was aggregated over the same users as the one of Zamani et. al., but the two datasets should be considered different. The data aggregation (for one user/journalist) is summarised in the following example:

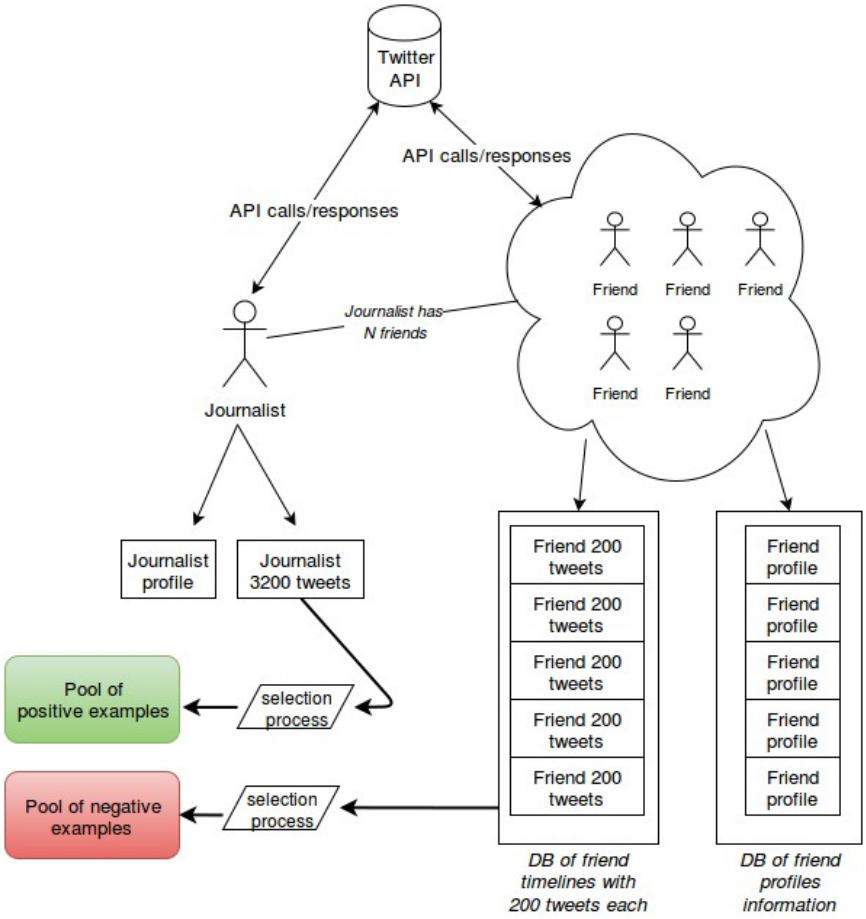

Figure 5: Example of data aggregation for a journalist

In the example above, we extract positive examples (retweets) from the journalist's timeline of 3,200 tweets, forming a positive example pool for this user.

We also extract negative examples from the timelines of the user's friends, containing their 200 most recent tweets. Despite the smaller number of tweets per timeline, the pool of negative examples is always much larger than the positive pool, because each journalist follows, on average, 400-500 friends. As a matter of fact, the largest positive pool observed, contained 2,700 instances, while the largest negative pool contained 180,000. In total, we aggregated a corpus of more than 12 million tweets (of which, 140,000 are retweets) and 63,800 users (journalists) and friends.

As transformation into features is a bottleneck in our system, in practice we

only transform and use the whole pool of positive examples of each user, but only a portion of the, much larger, negative pool, with random undersampling. This portion is approximately as large as the positive examples transformed, therefore the final, usable dataset for each user has a 50% ratio (approximately) of positive examples, as shown in Fig. 6.

It was, also, discovered that only 139, out of the 262 user datasets, contained more than 500 positive examples, as required by the second dataset principle we defined. As a result, we discarded 123 user datasets. We, also, discarded 17 more user datasets, through language filtering, subject to the third dataset principle we defined.

The **final dataset** contains positive and negative examples for **122 journalists**, divided into 122 concatenated subsets. Each user subset is also a concatenation of positive examples, followed by an equal number of negative examples (50/50 ratio). This ordering emulates realistic use cases, where a filter is continuously extended to include new users and the number of aggregated training instances is gradually increasing with the number of users.
Out of the 122 user datasets, we use the 80 first for training and development (validation, tuning) and the other 42 for the final evaluation of the best configuration of our system.

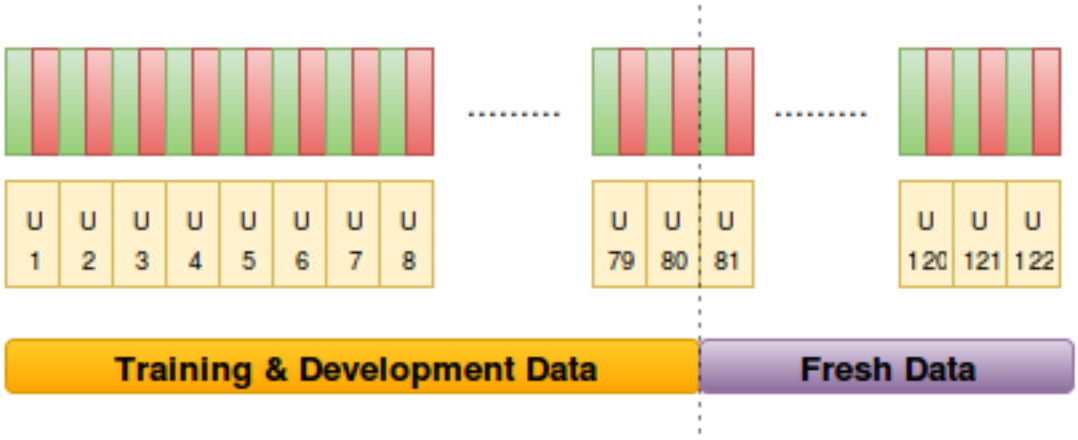

Figure 6: Final dataset of 122 journalists, partitioned

We note that one or more of our test users may encounter copies of the same tweet propagated to them via different paths. We distinguish between such paths and create distinct instances (feature vector representations), as demonstrated below, in Fig. 7.

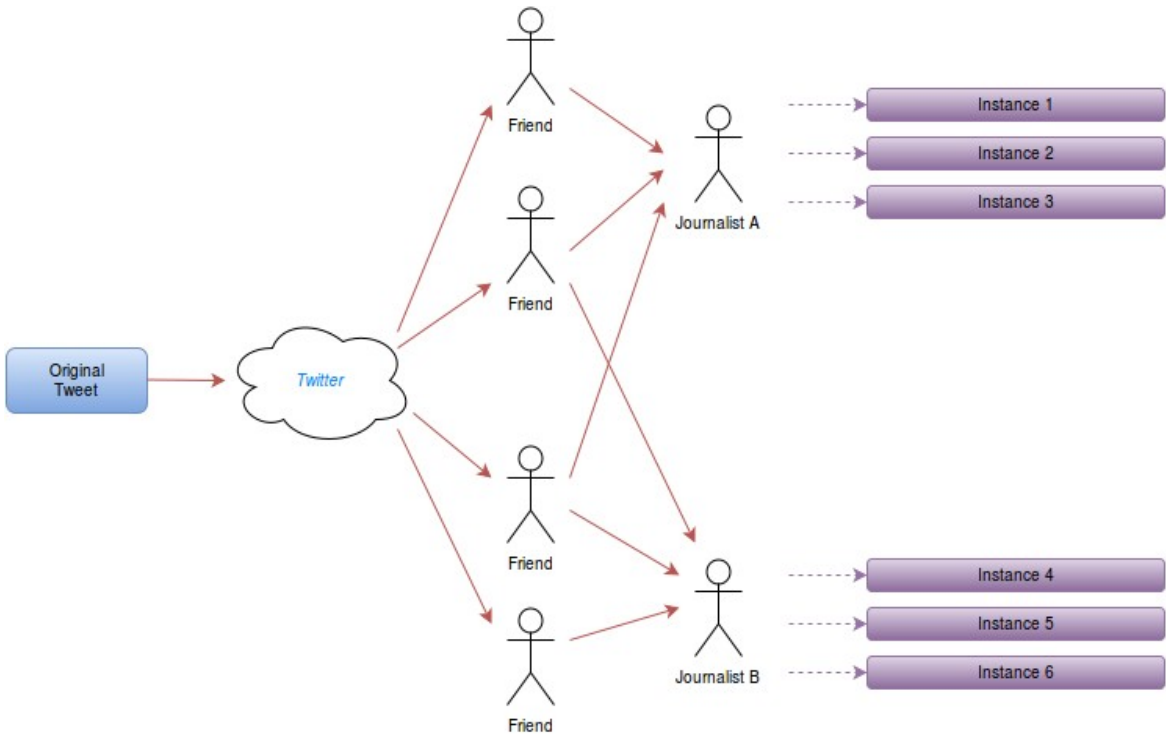

Figure 7: Different dissemination paths of the same tweet result in distinct instances

Interestingly enough, we discovered that instances stemming from the same original tweet, largely share the same class label, irrespectively of the user-aware and friend-aware feature values. We conclude that the same user assigns the same interest to the same tweet, no matter which friend the tweet came from[6]. Hence, including instances stemming from the same tweet in both the training and the test set might lead to over-optimistic evaluation results. To address this issue, we excluded from the test sets (in all of our experiments) instances stemming from the same tweets as instances of the training set, using the tweet ID number of the original tweet.

6 Note that the original tweet may have been written by the friend, or it may have been written by a third user and the friend may have retweeted it.

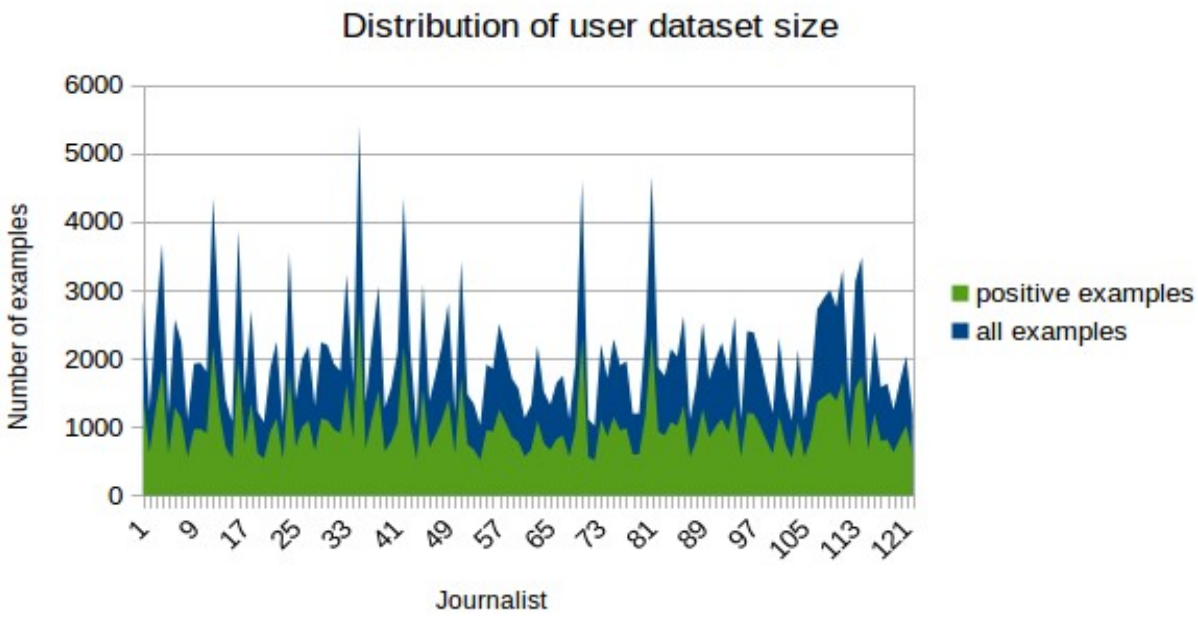

Chart 1: Final dataset statistics

Total number of users	122
Total instances in training+development	~160,000
Total instances in fresh test	~90,000
Total instances in dataset (in feature vector format)	~250,000

Table 2: Summarized statistics of the final dataset

4.1.3 Social influence data from Klout

Klout.com is a well-known provider of social influence indicators for users of any known social network. A Klout score is a single number ranging from 1 to 100. Other information, such as a user's main topics or topic-wide influence are also provided, though not used in the context of our study. Klout claims to collect and digest signals from multiple social networks that a user belongs to, apart from Twitter, in order to estimate social influence. While the methods used by Klout to provide influence scores are, to the best of our knowledge, not completely transparent, we nevertheless decided to include Klout scores in our dataset because of the service's popularity. For each Twitter user in our dataset (journalists and their friends), we stored their user influence scores and three deltas, denoting the changes of each Klout score over the last day, week and month, respectively.

We managed to find Klout scores for all 122 journalists and for the majority of their friends. For friends without Klout scores, we conservatively set their influence scores to 25, which is approximately the minimum value found in the dataset (an average user's Klout score is usually around 40).

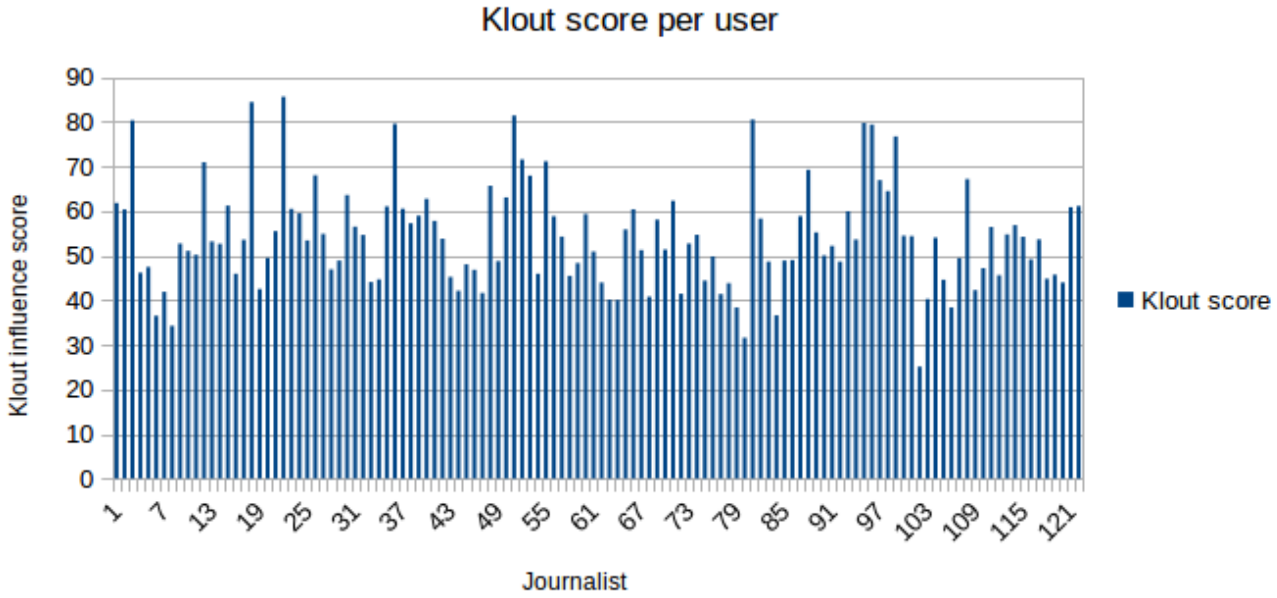

Chart 2: Distribution of Klout scores of journalists

4.2 Evaluation framework

The system is evaluated in two phases:

- During the development process, it is trained on a portion of the first 80 user datasets and tested on another portion of the first 80 user datasets. Evaluation in this phase drives decisions on system configuration (training data needs, model, features) and fine-tuning. The testing data (also called *development data*) used in this phase's evaluation should have properties as close as possible to the properties of data which the system will actually be employed to make predictions on. For instance, the ratio of positive examples in the test data, should loosely follow the ratio of the actual population. The test set of this phase can also be considered as having been used for system training (albeit in an indirect way), because it affects the choice of the system's configuration.
- After the development process, the system (its best configuration) is tested on fresh, unseen data, the 42 remaining user datasets.

In both phases, the evaluation framework performs two independent tasks in order to evaluate a system configuration:

1. Generating a learning curves pair (performance on train set/on test (or dev) set), in terms of a set-based metric, such as accuracy or F-measure of the positive class, at different amounts of training data. More specifically, during the development experiments, the curves we extract have nine data points each, for system training with 10%, 20%,...,90% of the training+development data (of the 80 first user datasets). Development testing is performed on a fixed test set (simple validation), which covers the remaining 10% of the training+development data. On the other hand, when evaluating the system on the fresh data (of the 42 other users), testing is performed on the entire fresh dataset of the 42 users. Due to long execution times, it was not feasible to perform cross-validation. Since the datasets are ordered by user, an increasing training set means that the system is trained on data of more users. Learning curves allow us to diagnose a plethora of system weaknesses, such as overfitting, limited search space, model oversimplicity and training data insufficiency.
2. Plotting Mean Interpolated Precision at different Recall levels ($MIP(r)$) of the positive class, at different amounts of training data. Five curves are shown in each diagram, one for each amount of training data (20%, 40%, 60%, 80%, 100%) and each curve has eleven data points, showing MIP at each Recall value (0,0.1,0.2,...,0.9,1). During the development phase, each curve is constructed by taking into account the system performance when asked to provide ranked results (a list of tweets ranked by decreasing retweet likelihood) for each one of the 80 development users. This is implemented through an 80-fold cross-validation test, with leaving one user out of the training set at each iteration. An iteration yields 11 values of Interpolated Precision at different Recall levels ($IP(r)$) for the user left out. These are then averaged over all users to obtain the $MIP(r)$ scores of the system. During the second phase, when evaluating on the fresh test data, the system is trained using the 80 first user datasets and consequently it is asked to provide ranked results for each one of the 42 test users. Similarly, an iteration yields 11 $IP(r)$ values for the user left out and these are then averaged over all 42 users to obtain the $MIP(r)$ scores. By using this rank-based evaluation, we obtain a better impression of the system performance in an actual use case (personalised tweet ranking) and useful insights when realistic, user-provided requirements are applied onto the system (e.g. “I want to see at least 80% of the interesting tweets”, which means minimum recall=0.8, or “I want to see no more than 20% uninteresting tweets in my timeline, at a given moment”, which means minimum precision=0.8). The existence of five curves per system, for different amounts of training data, serves as another way to spot training data insufficiency issues.

4.3 Evaluation measures

4.3.1 Set-based measures

A set-based metric evaluates a binary prediction system, in terms of Type I and Type II errors on the class we focus on (here, this is the positive class), using unordered sets of instances. Instances are predicted to belong to either class, based on whether they are over or under a probability threshold, without taking into account the classifier's confidence. While thresholds usually vary to accommodate different needs in specificity/sensitivity, in our experiments the threshold is set to 0.5 when using set-based evaluation measures. Consequently, an instance predicted to be retweeted with a likelihood of at least 50% will be allocated to the positive class. Due to the unordered nature of the metric, an instance allocated to the positive class with 51% confidence is treated no differently than an instance allocated to the positive class with 99% confidence.

We define, for the *positive class*:

- **true positives (tp):** number of test instances predicted to belong to the positive class and actually belonging to the positive class
- **true negatives (tn):** number of test instances predicted not to belong to the positive class and actually not belonging to the positive class
- **false positives (fp):** number of test instances predicted to belong to the positive class, but actually not belonging to the positive class (type I error)
- **false negatives (fn):** number of test instances predicted not to belong to positive class, but actually belonging to the positive class (type II error)

Using the quantities above, we further define:

$$Precision = \frac{tp}{tp+fp} \qquad F1 = 2\frac{Precision \cdot Recall}{Precision + Recall}$$

$$Recall = \frac{tp}{tp+fn} \qquad Accuracy = \frac{tp+tn}{tp+tn+fp+fn}$$

In our experiments, we used Accuracy and/or F1 when plotting learning curves. Accuracy often distorts conclusions under class imbalance conditions. For instance, given a test dataset where 95% of examples belong to the negative class and a baseline majority classifier which allocates all instances to the negative class, accuracy will be 95% because of the very high number of true negatives. On the contrary, F1 (harmonic mean of precision and recall with equal weight on both) focuses on one class (in our case, the positive class) and the F1 score would be undefined (intuitively, we can consider the F1 score to be 0% in such case).

4.3.2 Ranked-based measures

A rank-based metric evaluates the system by taking into account the quality of ranked results, for instance a list of tweets ranked by decreasing likelihood to be retweeted by a specific user. Stemming from Information Retrieval (IR), these metrics do not utilise threshold-based hard decisions of the classifier. For instance, if a system manages to return all actually positive examples in higher positions of the ranked list than the negative ones, but gives very low (below the set-based threshold) confidence for all the examples, then the system successfully serves its purpose, but accuracy is very low. We believe that rank-based metrics reveal a better picture for system's performance.

Based on Information Retrieval concepts, we devised the mapping:

Information Retrieval	**Tweet recommendation**
Query	Twitter user
Document	Tweet
Relevant document	Tweet which the user would retweet
Non-Relevant document	Tweet which the user would not retweet
Query-document similarity	Probability that the user would retweet the tweet (positive class confidence)

Table 3: Problem mapped to Information Retrieval concepts

Based on the mapping above, we can put forward an alternative definition for our system's intended task:

→*The system should provide a ranked list of tweets, with the most "relevant" to the user being as high in the list as possible.*

Consequently, based on the k -user testing explained in Section 4.2 (in an IR system, its counterpart would be the submission of k test queries), we do as follows, *for each one of the development/test users*:

1. We rank each tweet T_i of the iteration's test/development set (positive and negative examples of the test user) by descending classifier confidence that it would be retweeted by the user (probability that it belongs to the positive class).
2. We then mark each tweet T_i with its true label, which is whether the test user actually retweeted the tweet ("relevant"), or not.
3. Afterwards, we compute the precision P_i and the recall R_i up to each position i of the ranked list, as follows:

$$Rels = |\text{actually relevant tweets for the test user}|$$

$$P_i = \frac{|\text{relevant tweets for the user up to index i}|}{i}$$

$$R_i = \frac{|\text{relevant tweets for the user up to index i}|}{Rels}$$

A visualised example of the ranked list, enriched with the computed metrics at each position, is as follows:

Rank (i)	True label	Precision	Recall
1	*R(elevant)*	*1.0*	*0.11*
2	*N(on relevant)*	*0.50*	*0.11*
3	*R*	*0.66*	*0.22*
...	*...*	*...*	*...*
25	*R*	*0.36*	*1.0*

Table 4: Ranked list example (for a test set of 25 tweets)

We then compute the Interpolated Precision at 11 Recall values, $IP(r)$, where $r \in \{0.0, 0.1, 0.2, ..., 1.0\}$, using the rule $IP(r) = max_{i:R_i \geq r} P_i$, i.e., the interpolated precision at a certain recall level r is the maximum observed precision of all ranks having recall equal to or greater than r .

Interpolated precision removes jiggles typically present in a normal precision-recall curve plotted using all pairs of precision-recall values of Table 4, although it may overestimate precision at some recall levels. For related discussion, consult Chapter “Evaluation in Information Retrieval” of Manning et al. [Manning, Raghavan, Schutze, 2008].

Consequently, *by averaging over all test users*, we obtain the Mean Interpolated Precision at 11 Recall values, $MIP(r)$, defined as:

$$MIP(r) = \frac{1}{K} \sum_{k=1}^{K} IP_k(r)$$

, where K is the number of test users ($K = 80$ in the development phase, $K = 42$ when testing on fresh test data).

A visualised summarisation of $MIP(r)$'s calculation, as well as the related average interpolated precision (AIP_k) and mean average interpolated precision

($MAIP_{system}$), which we did not use, is shown in Fig. 8. $MIP(r)$ is the function plotted in the precision-recall trade-off curves, generated by the evaluation framework.

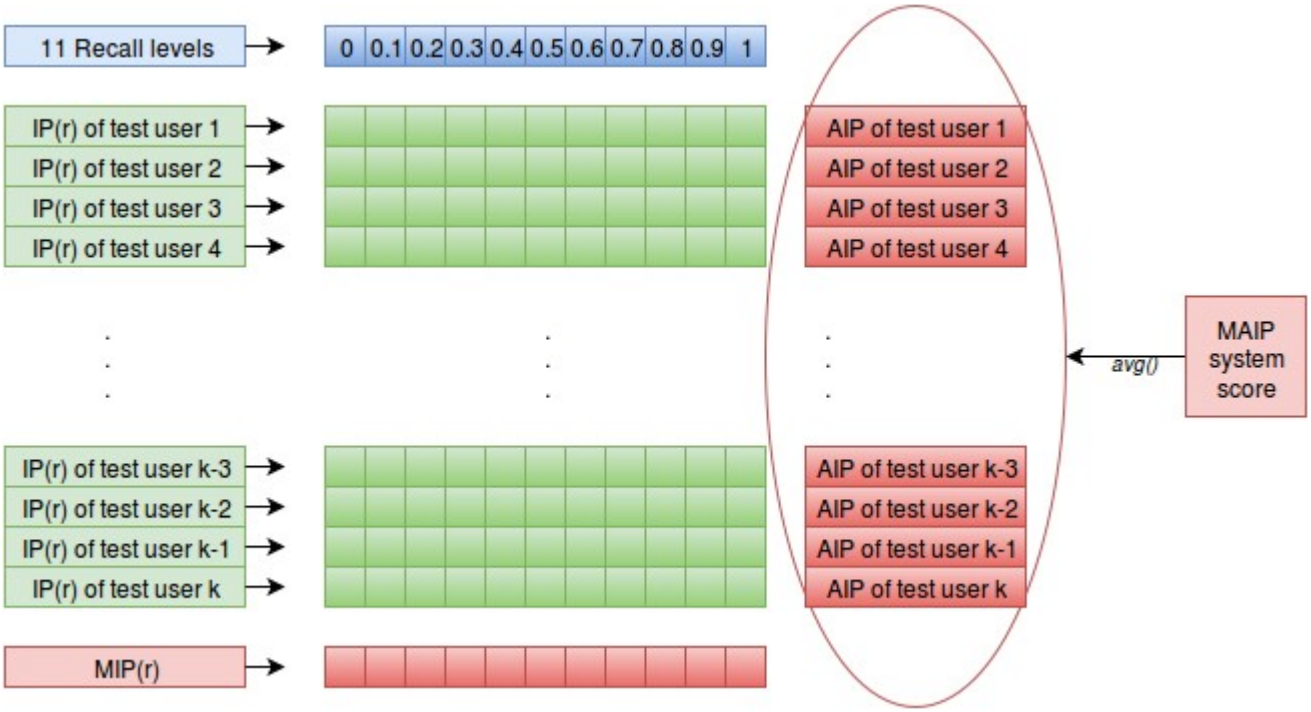

Figure 8: Calculating rank-based metrics in a multi-user validation (cross-validation in the development phase, simple validation in the final testing phase)

4.4 Preliminary experiments

In a first set of development experiments, we aimed to answer the following questions:

- How many (approximately) training examples does the system need to predict reasonably well which tweets will be retweeted?
- What should the positive examples ratio be in the training/development sets for the system to better predict the retweets?

The first question can be partially answered through the standard learning curves we construct. Each point in a learning curve is the F1 score as larger training sets are used. In our experiments, we show the F1 score on 10% of the training+development data, 20%, ..., 90% of the training+development data. The reader is reminded that the last 10% of the training+development data is the test subset (thereafter called simply *development set*). The percentages of training data we use can be loosely mapped to absolute numbers of training tweets as follows:

10%	20%	30%	40%	50%	60%	70%	80%	90%
15,600	31,200	46,800	62,500	78,100	93,700	109,300	125,000	140,600

Table 5: Relation between percentages and numbers of training instances used in learning curves

Nevertheless, we also experimented with much fewer training examples in total. In this "few data" training mode, there is a different mapping:

10%	20%	30%	40%	50%	60%	70%	80%	90%
1,500	2,800	4,400	6,000	7,300	8,900	10,500	12,000	13,600

Table 6: Relation between percentages and numbers of training instances in "few data" training

Below, we compare the two training modes in terms of F1 score in the positive class, for different amounts of training data, using the Logistic Regression implementation of Weka[7], with default parameter values (presented in Section 4.5). In these experiments, the negative training and development (test) instances were randomly undersampled to obtain a 50% ratio of positive instances.

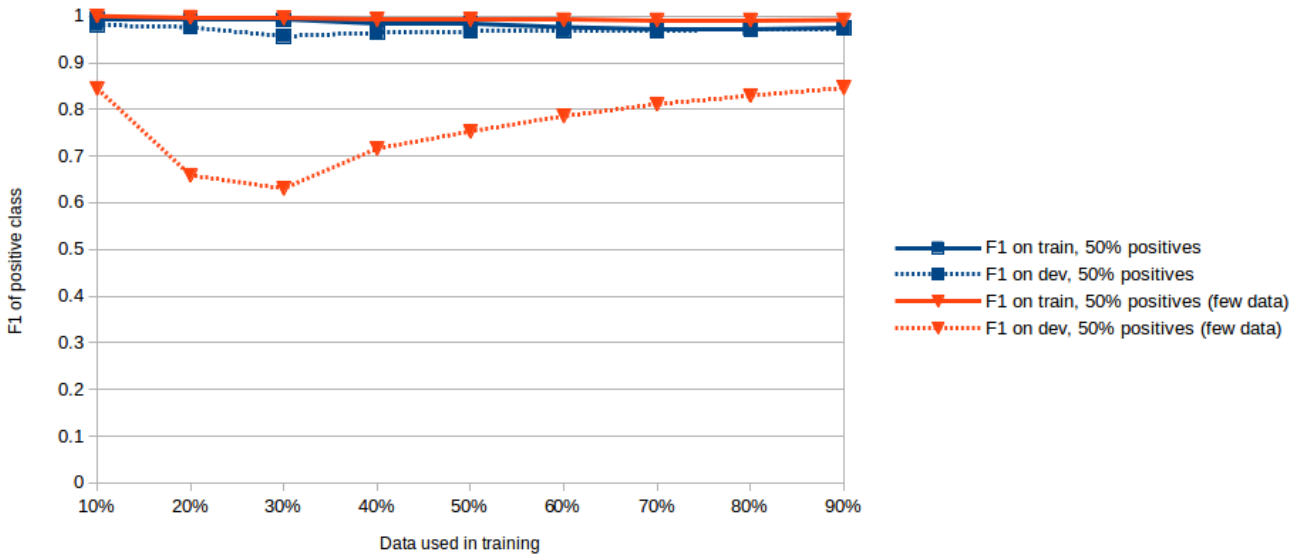

Chart 3: F1 of the positive class for different sizes of the training set, using Logistic Regression and a 50% ratio of positives in the training and development sets

From the plot above we understand that when the system is trained on very few data, it tends to overfit the small training set, leading to lower F1 score on test (dev) instances. Therefore, adding more training data boost the system's performance, as both the change rate in the red on-development curve and the

7 http://weka.sourceforge.net/doc.dev/weka/classifiers/functions/Logistic.html

height of the blue on-development curve reveal. Nevertheless, it seems that a few thousand instances sufficiently train the model up to most of its prediction ability. This may explain the relatively flat blue F1 curves, where the full training dataset is used, when contrasted to the red curves, which offer a more fine-grained picture on the first few thousand training instances.

The second question can be answered by performing experiments using varied ratios of positive examples in the train and/or development set. In the development set, this ratio should generally be as close as possible to the ratio of positive examples in the population (less than 5%, as estimated in our collected data) to be close to realistic filter application conditions. On the other hand, we are generally free to select the positives ratio in the training set. In the system setups of the experiment below, we use different ratios of positive examples (applicable both to the train and development set), to showcase how accuracy is affected.

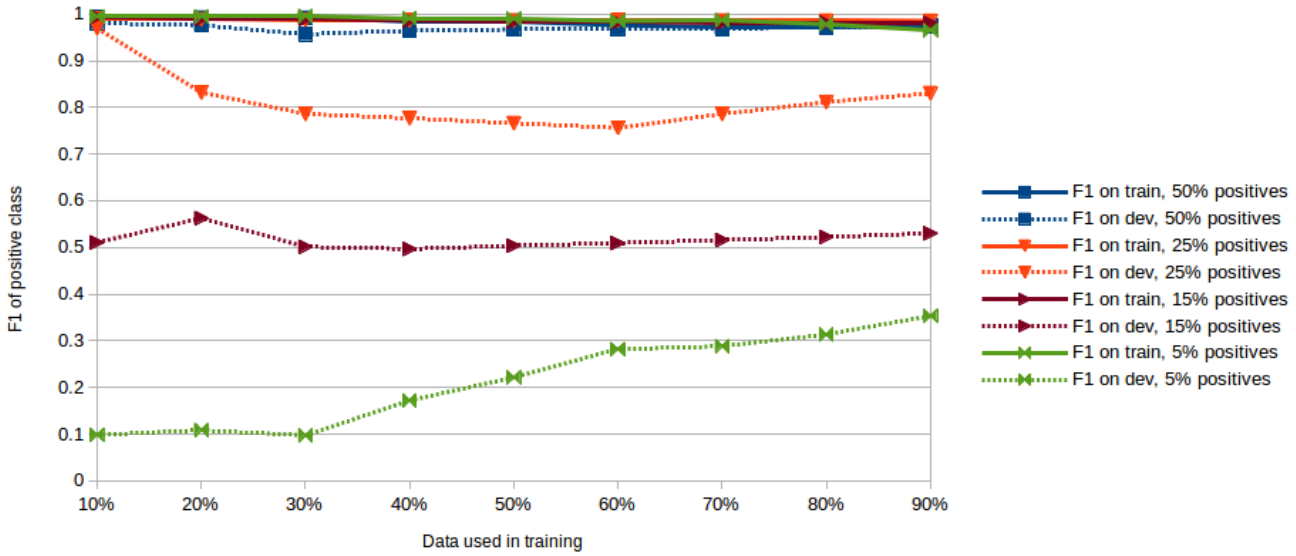

Chart 4: F1 of the positive class for different sizes of the training set, using Logistic Regression and different ratios of positives in the training and development sets (same ratio in each pair of training and development sets)

Overall, the task becomes more difficult when lower positive ratios are employed. Nevertheless, still acceptable (and increasing) levels of F1 are observed when using the most realistic ratio of 5% positives. The acceptable level of performance at this ratio is also confirmed by the rank-based evaluation plot (Chart 5), where high precision is achieved at most recall levels.

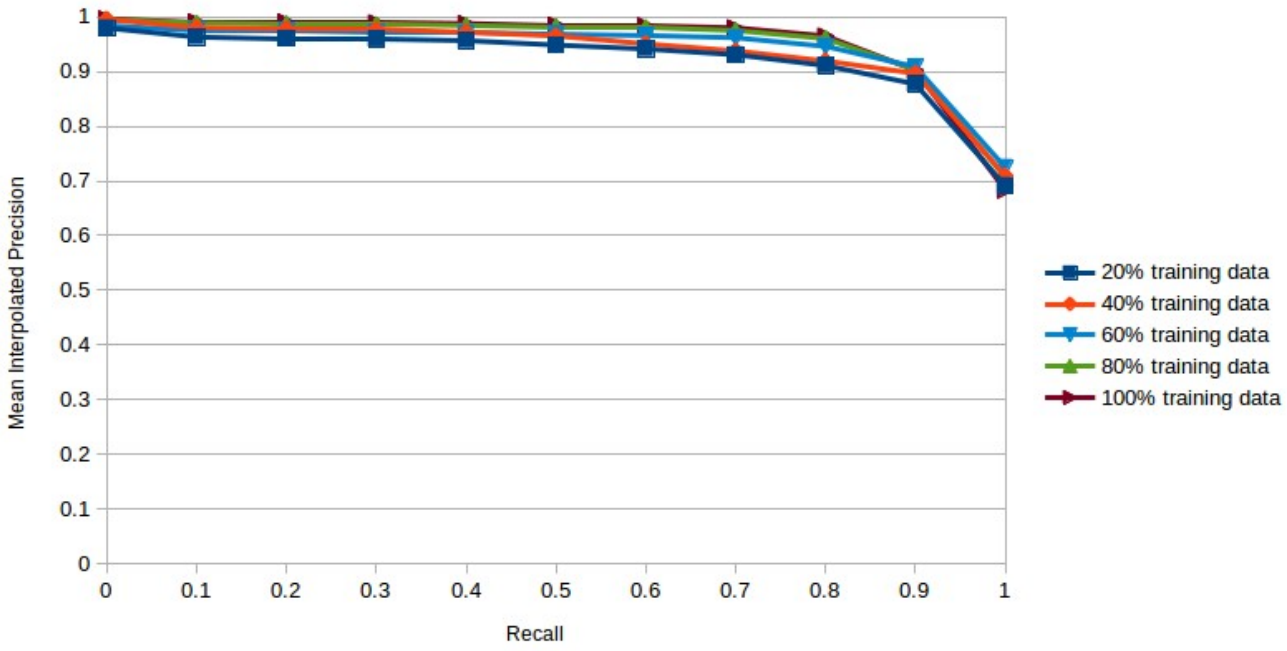

Chart 5: Mean Interpolated Precision at different recall levels, for different sizes of the training set, using Logistic Regression and 5% positives in the training and development sets

In the previous experiment we set the positives ratio in both the training and the development set to 5%. In the subsequent experiment, we set the ratio to 50% in the training set and to 5% in the development set. This setup keeps the development set realistic enough, while using the most favourable ratio of positives in the training set.

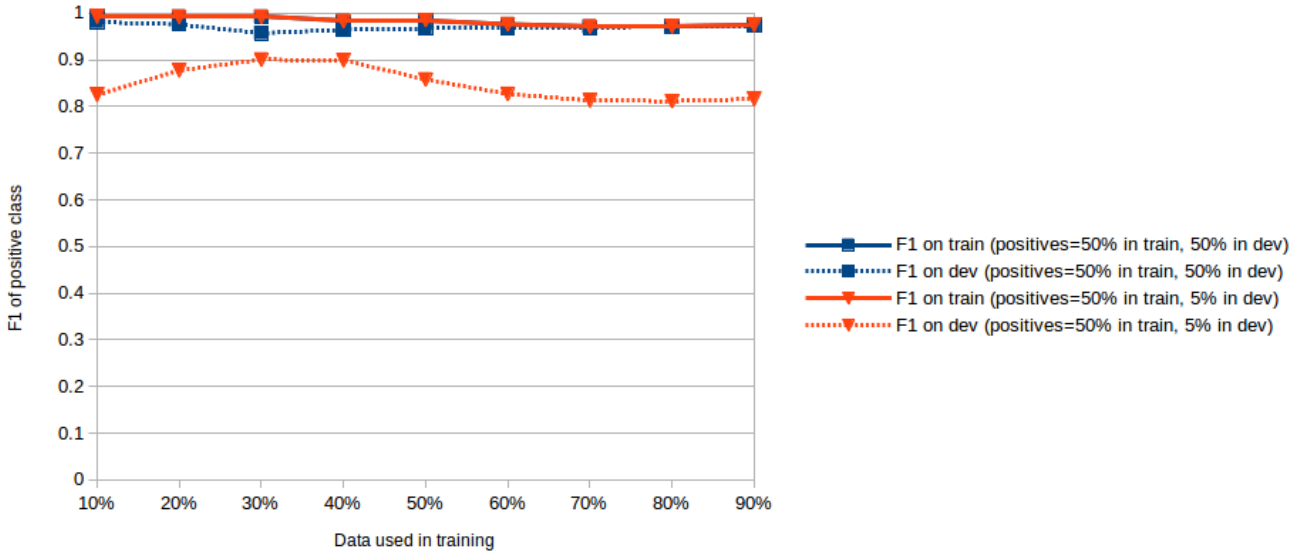

Chart 6: F1 of the positive class for different sizes of the training set, using Logistic Regression, 50% positives in the training set and 5% or 50% positives in the development set

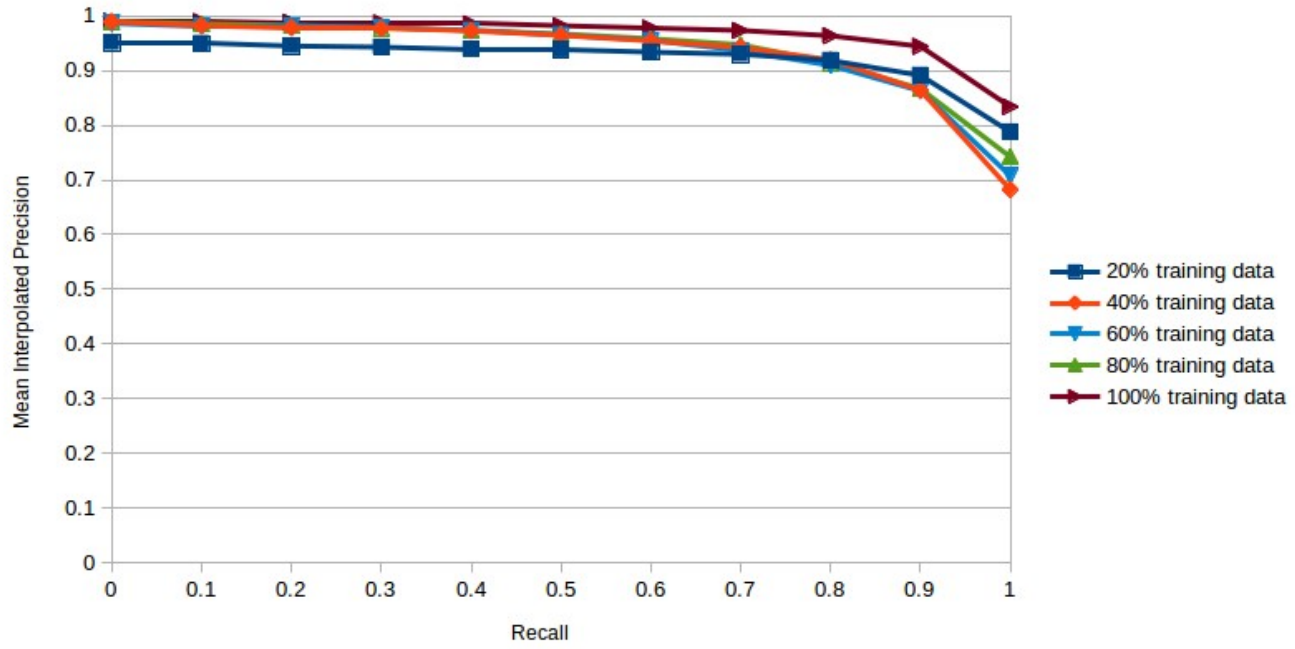

Chart 7: Mean Interpolated Precision at different recall levels, for different sizes of the training set, using Logistic Regression, 50% positives in the training set and 5% positives in the development set

From Chart 6 we can verify that it is mostly the ratio of positive examples in the *training* set that affects the system's performance, while changing the ratio of positives in the development set introduces a smaller difference. Therefore we

can pick the best ratio for the training set, 50%, and keep development set at the realistic 5%. This setup's rank-based evaluation (Chart 7) also yielded a quite promising result.

4.5 Experiments for model selection on development data

In this experiment, based on our previous experience [Vougioukas, 2014], we examined whether a decision tree-based model ID3 (J48 Weka implementation[8], with default parameter values) could yield better results than the Logistic Regression classifier we used in the previous experiments. We used the following settings:

- Logistic Regression
  - Ridge: 10^{-8} (default in implementation)
  - Max iterations: unlimited, until convergence
- J48
  - Tree pruning: no
  - Minimum number of instances per leaf: 2
  - Allowed split type: binary

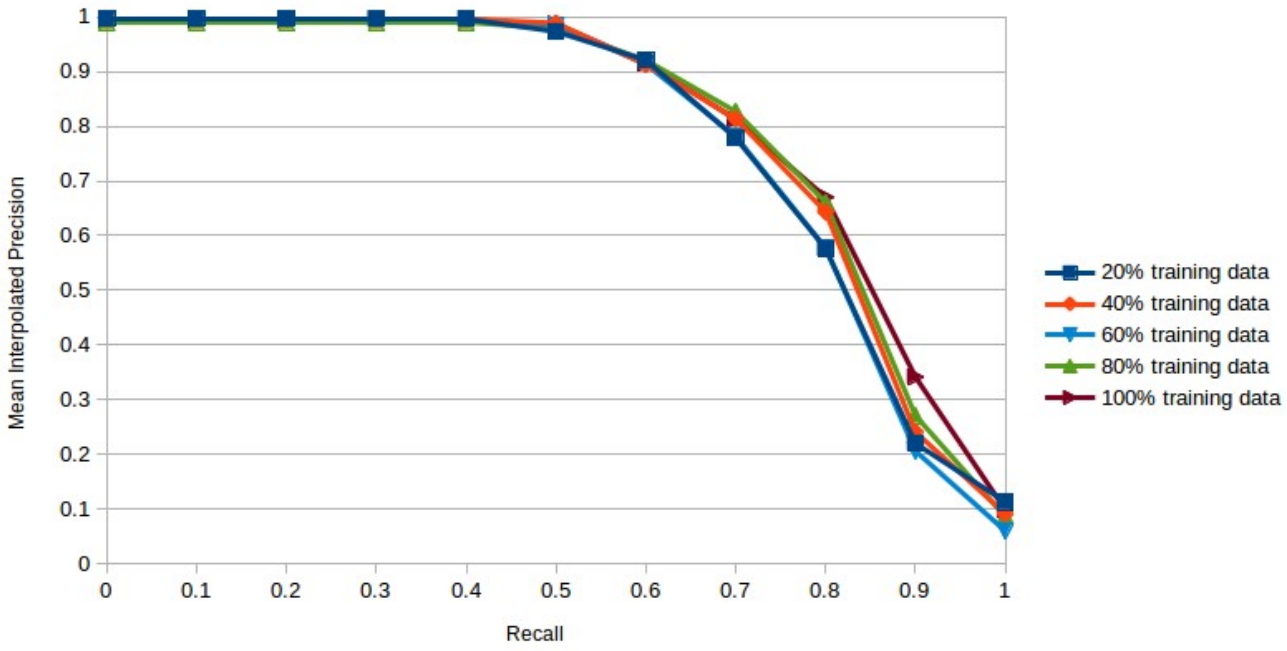

Chart 8: Mean Interpolated Precision at different recall levels, for different sizes of the training set, using J48 and 5% positives in the training and development sets

8 http://weka.sourceforge.net/doc.dev/weka/classifiers/trees/J48.html

Comparing the rank-based evaluation curves in Chart 5 (Logistic Regression) and Chart 8 above, we understand that J48 decision trees achieve, on average, lower precision at most recall levels, when compared to the Logistic Regression classifier. This is especially true in higher recall levels, which makes it much more possible for the filter user to miss important/interesting tweets. In the ranked list use case, one would have to have to scroll through many more unimportant tweets to see all the important ones, if J48 decision trees were employed. Therefore, Logistic Regression has a clear advantage in the critical area of high recall levels.

As a result, we chose to continue using the Logistic Regression classifier. Other advantages from its use include its simplicity, efficiency, ability to expose the logistic function coefficients (weights attributed to each feature in training) and ability to tune ridge, according to different regularisation needs. On the other hand, it may be possible to improve J48's performance by using limited maximum depth and tree pruning, in an effort to prevent learning very deep trees, which implies model overfitting. Despite not using J48, its tree visualisations (Fig. 9) offer useful knowledge about the features. For instance, from the tree below we understand that some of the features are very correlated with the class label, which they can predict with very little contribution from other features. Moreover, the tree informs us about how the feature value ranges are split, in order to discriminate between examples of the two classes.

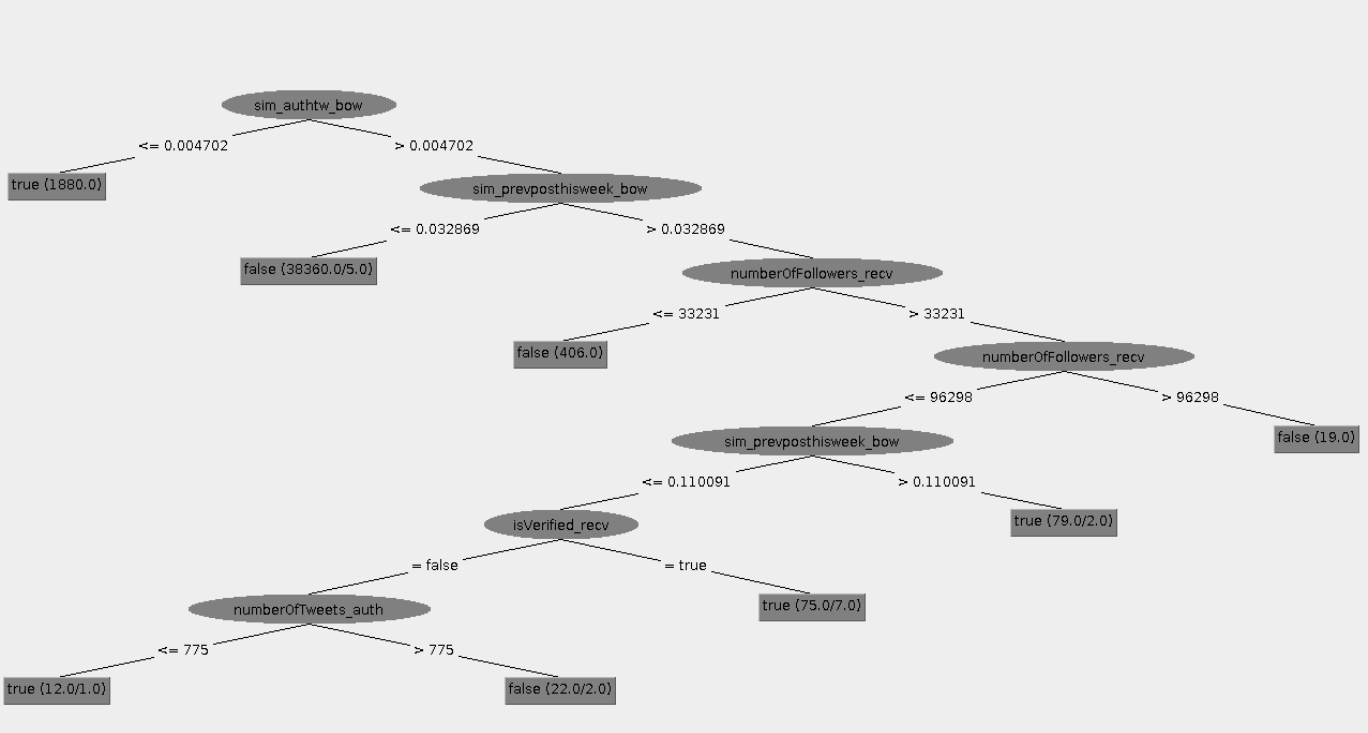

Figure 9: Decision tree produced by J48 using approximately 40,000 training instances and 5% positives in the training dataset

4.6 Experiments for feature set selection

To begin with feature selection, we ranked all fifty features by decreasing Information Gain (IG). IG estimates a feature X 's correlation with the class label C , independently of other features, over a large example set. This is a standard and quite efficient method, especially when used in conjunction with a 10-fold cross validation, as we did.

We define:

$H(C)=-\sum_{c} P(C=c)\cdot\log_2 P(C=c)$, entropy of class label C with values c

Entropy measures the uncertainty of a random variable's value and (in binary variables) ranges from 0 (full certainty) to 1 (full uncertainty).

The IG of a random variable X with respect to a random variable C expresses the expected decrease in C 's entropy when the value of X is known. It is defined as:

$IG(C,X)=H(C)-\sum_{x} P(X=x)\cdot H(C|X=x)$, where x are the possible values of X

and:

$$H(C|X=x)=-\sum_{c} P(C=c|X=x)\cdot\log_2 P(C=c|X=x)$$

Based on IG, the features are ranked as follows. Estimations of IG values and ranks were taken over ten folds, therefore the averaged values are presented:

Feature	Avg. IG	Avg. rank	Feature	Avg. IG	Avg. rank
FT43	0.847+-0.001	1	FT6	0.004	26
FT10	0.83+-0.001	2	FT46	0.002	27
FT23	0.706+-0.013	3.7+-1	FT9	0.002	28
FT24	0.706+-0.009	4.3+-1	FT3	0.001	29
FT22	0.693+-0.022	4.8+-1.17	FT18	0.001	30
FT21	0.689+-0.013	5.2+-0.6	FT8	0.001	31
FT16	0.646+-0.008	7.1+-0.3	FT4	0.001	32.1+-0.3
FT14	0.629+-0.009	7.9+-0.3	FT50	0.001	32.9+-0.3
FT15	0.369+-0.002	9	FT40	0	34
FT17	0.359+-0.002	10	FT48	0	35
FT13	0.315+-0.001	11	FT7	0	36

FT19	0.224+-0.003	12	FT38	0	37
FT11	0.199+-0.001	13	FT20	0	38.3+-0.46
FT41	0.151+-0.001	14.2+-0.4	FT49	0	38.7+-0.46
FT44	0.149	14.8+-0.4	FT31	0	40.3+-0.46
FT45	0.132	16	FT29	0	40.7+-0.46
FT5	0.108	17	FT47	0	42
FT42	0.095+-0.001	18	FT35	0	43.1+-0.3
FT30	0.067	19	FT33	0	44.4+-1.2
FT36	0.047	20	FT34	0	44.8+-0.6
FT12	0.015	21	FT32	0	46.3+-0.9
FT1	0.01	22	FT26	0	46.9+-0.3
FT37	0.007	23	FT27	0	47.7+-0.9
FT39	0.005	24	FT28	0	48.8+-0.6
FT2	0.004	25	FT25	0	50

Table 7: Ranking of features by decreasing Information Gain

From the ranking above, we see that the two top features are always first and second in IG, respectively, the following four fluctuate in positions 3-6, while most of the rest have relatively fixed positions. Moreover, the top eight features have very high IG scores (over 0.62), followed by ten features with high IG (0.10-0.37), while the bottom seventeen features scored zero IG in all ten folds.

Below, we test the system using the top-10, top-20, top-30, top-40 and all the features. We evaluate the different setups using learning curves (as we want to detect change in overfitting behaviour) in terms of F1 of the positive class.

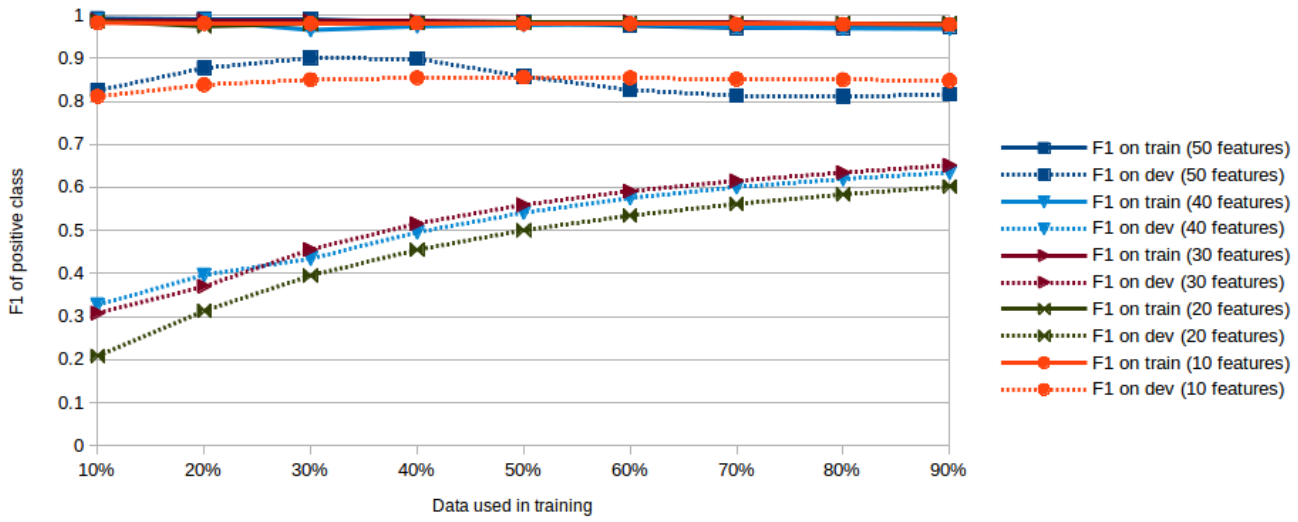

Chart 9: F1 of the positive class, using Logistic Regression, varying the number of top features used

From the plot above, we may comment that:

- Based on the on-train curves and their distances from the on-development corresponding curves, our system faces serious overfitting for reasons that need to be examined. We would expect the performance on the training dataset to decrease as more data are added and/or features are removed.
- Based on the on-development curves for the top 20, top 30 and top 40 features, it seems that the aforementioned issue may be compensated by an increasing performance on development data, when more training data are used. Strangely enough, a much higher on-development F1 score is observed, yielding much flatter curves when the top 10 or all features are used, even with very few training data. When using the top 10 features, this behaviour may be attributed to less overfitting, but we have no obvious explanation for the high on-development performance when using all the (50) features.
- All the observations mentioned above need further investigation, preferably combined with a qualitative analysis of the features, in order to establish sound conclusions about the best features to use.

We chose to use the top 20 features in the following experiments, rather than the perhaps too optimistic top 10 ones. Judging from the rank-based system evaluation presented in Chart 10, we can still achieve acceptable performance with the top 20 features, especially when using over 40% of the training data (~60,000 examples). This is clearer in Chart 11, where we focus on the critical area of high recall levels (above 0.6).

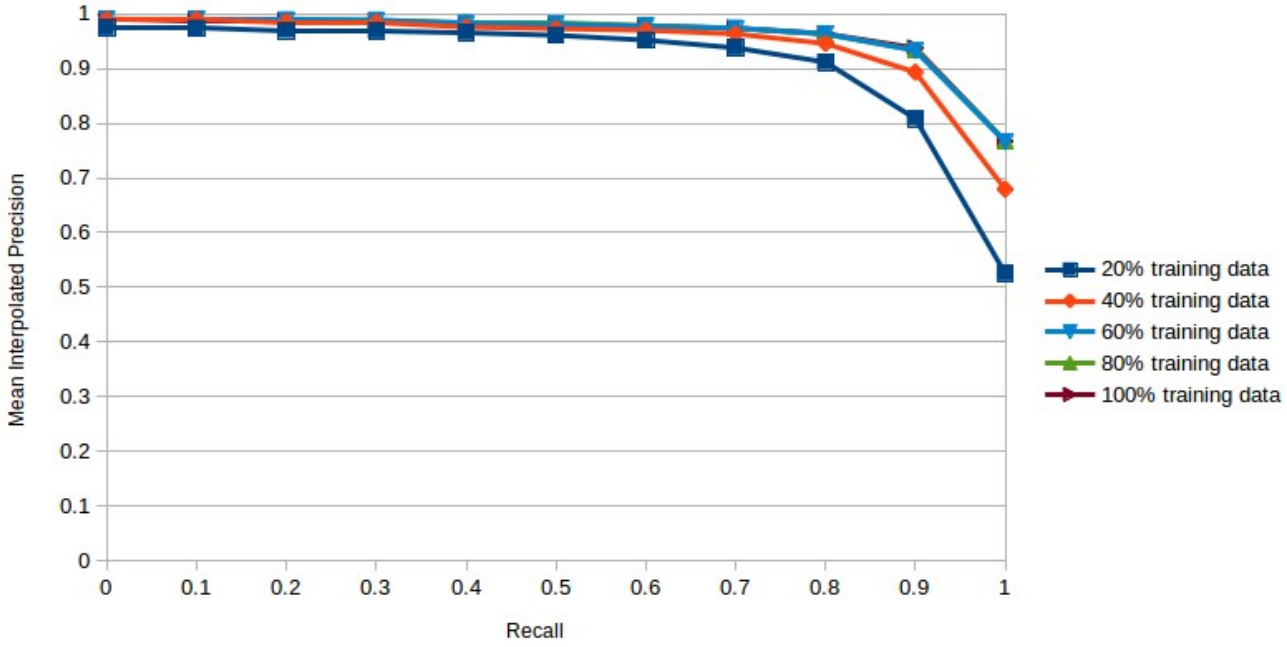

Chart 10: Mean Interpolated Precision at different recall levels, for different sizes of the training set, using Logistic Regression, 50% positives in the training set, 5% positives in the development set and the top 20 features

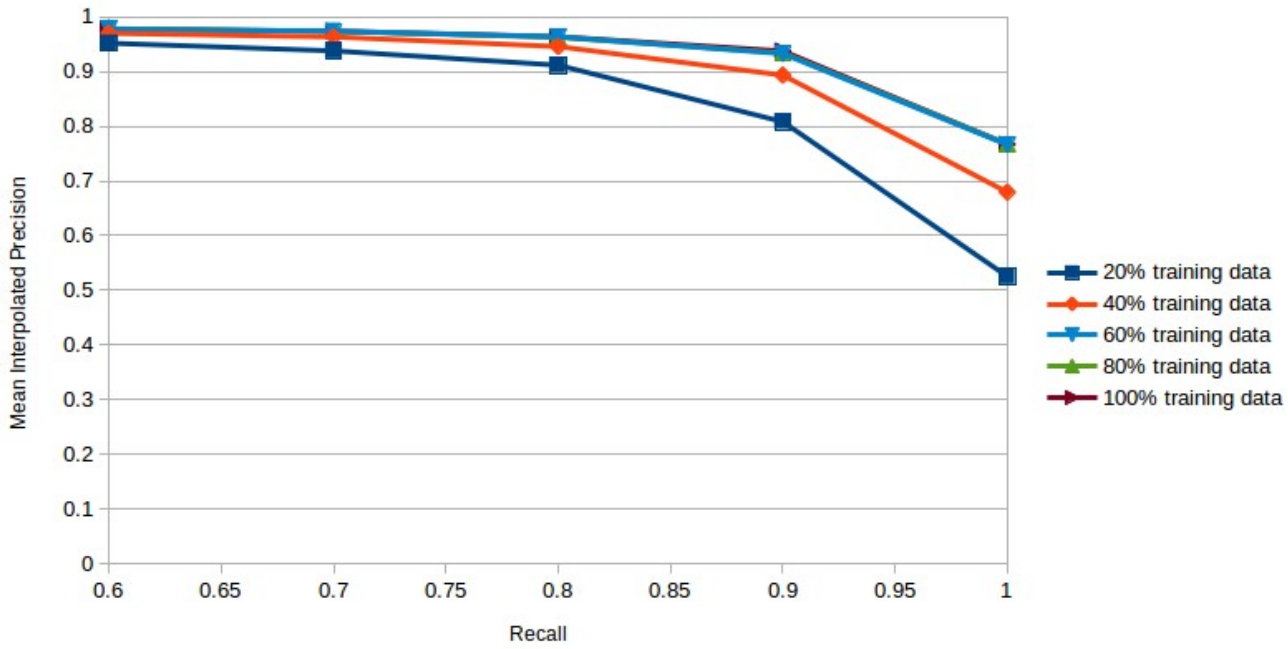

Chart 11: Mean Interpolated Precision at different recall levels, for different sizes of the training set, using Logistic Regression, 50% positives in the training set, 5% positives in the development set and the top 20 features
(zoomed at high recall area)

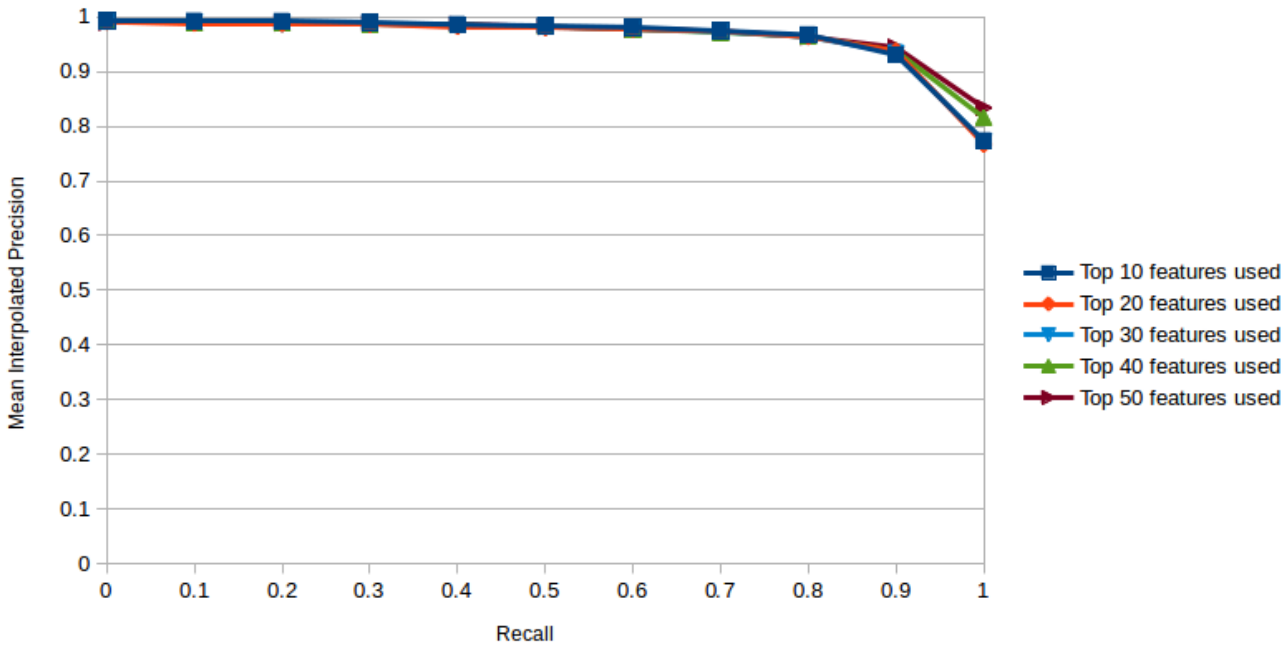

Chart 12: Mean Interpolated Precision at different recall levels, for 100% of the training set, using Logistic Regression, 50% positives in the training set, 5% positives in the development set and varying the number of top features used

4.7 Evaluation of the final system on fresh test data

Based on the previous experiments, we propose the final configuration of the system, which is as follows:

Prediction type	Binary
Label set	{negative,positive}
Prediction model	Logistic Regression classifier
Feature set	Top 20 features, as listed in Table 7
Positive examples ratio in training set	50%
Positive examples ratio in development/test set	Indifferent, defined by distribution in population (here set to 5%)

Table 8: Properties of final proposed system setup

Below, we evaluate the final system setup configuration on the 42 held-out user datasets (fresh data), unseen during the training and development. Rules to avoid including instances stemming from the same original tweet in both training and test set are also in place here.

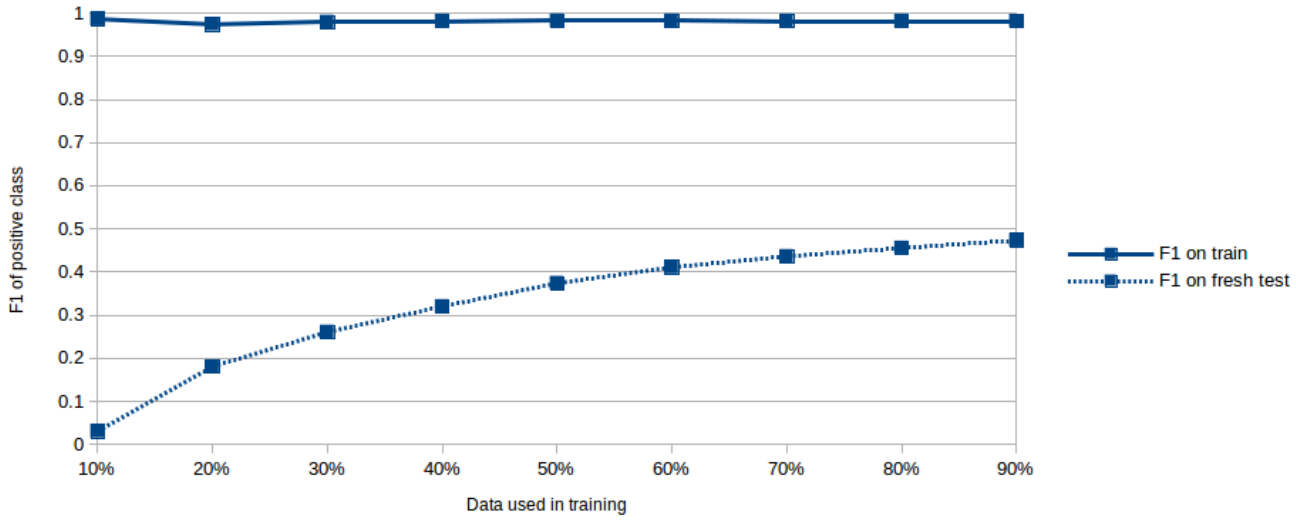

Chart 13: F1 of the positive class, using the fresh test data and the system configuration described in Table 8

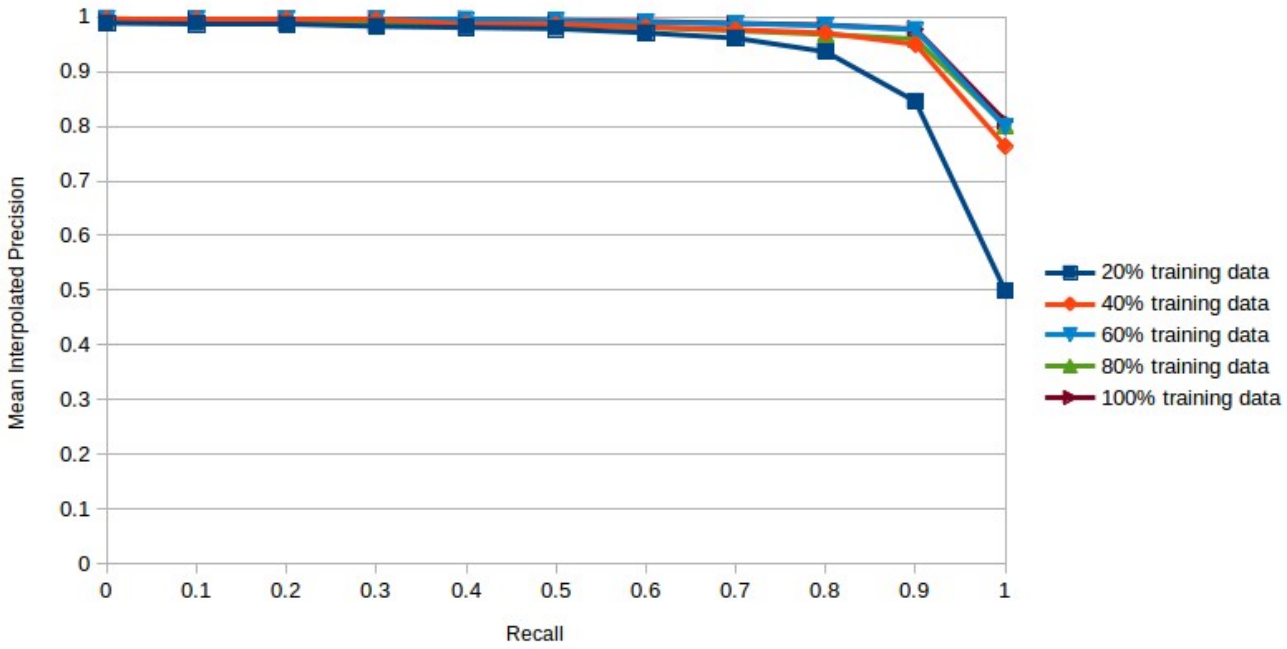

Chart 14: Mean Interpolated Precision at different recall levels, for different sizes of the training set, using the fresh test data and the system configuration described in Table 8

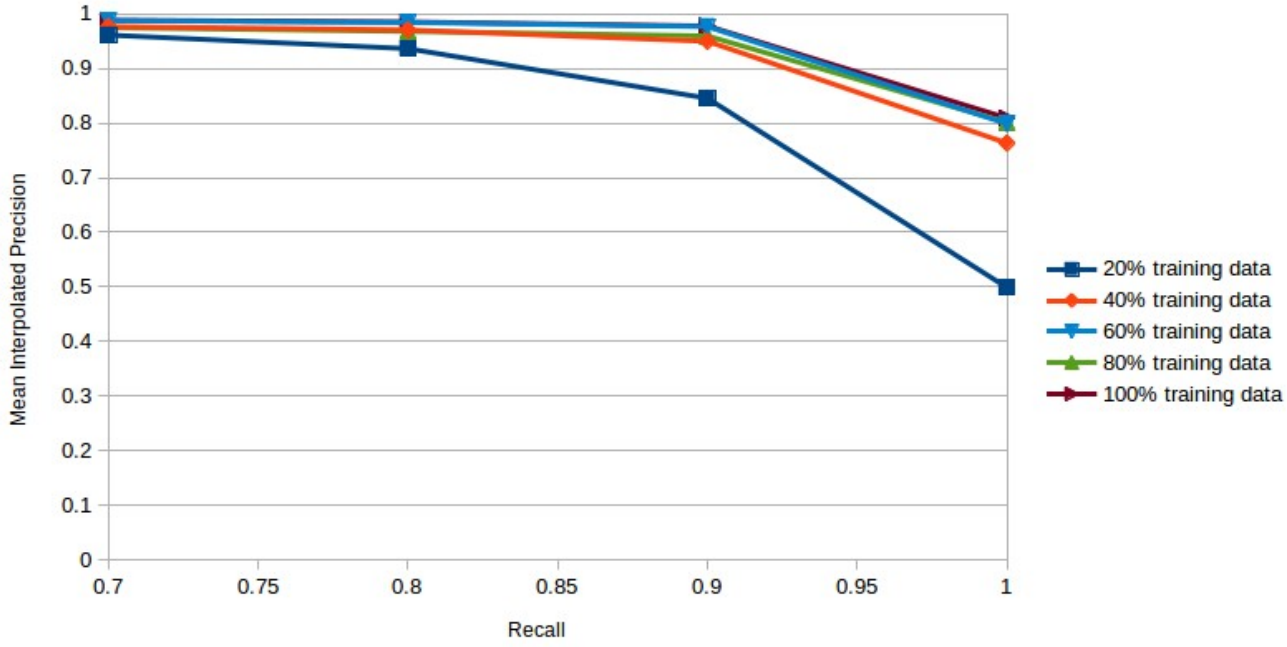

Chart 15: Mean Interpolated Precision at different recall levels, for different sizes of the training set, using the fresh test data and the system configuration described in Table 8 (zoomed at high recall area)

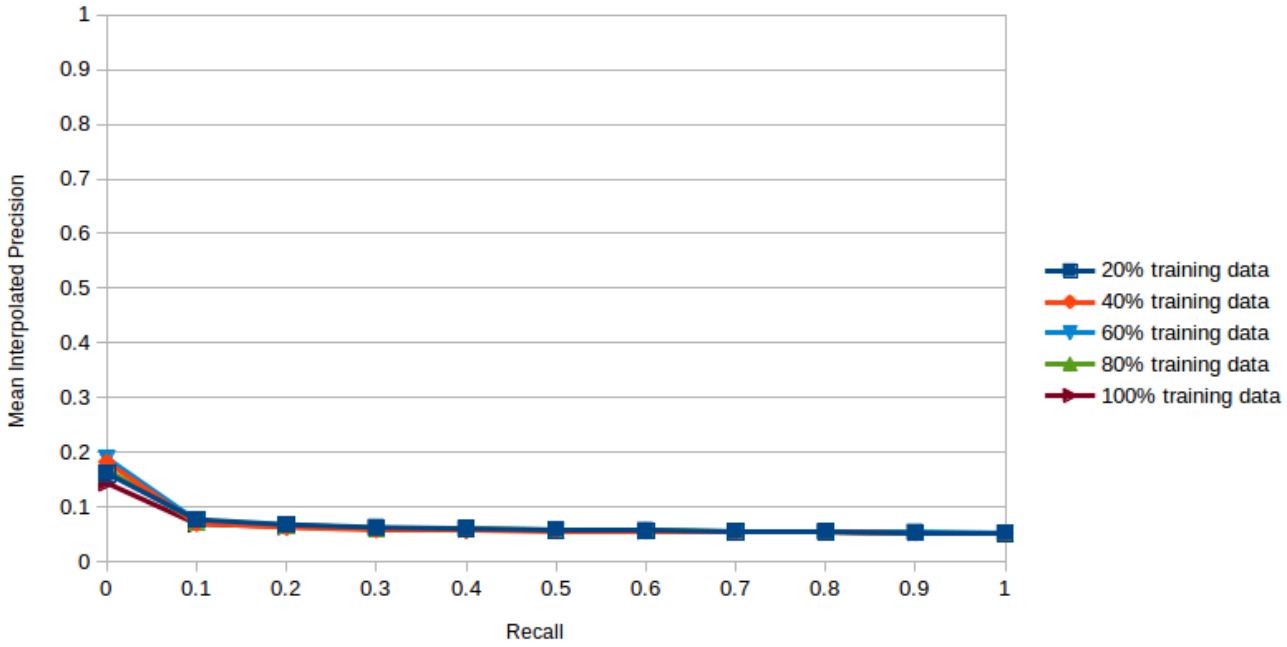

Chart 16: Mean Interpolated Precision at different recall levels, using the fresh test data and a naïve system with a random tweet ranker, which we use as a baseline for comparison

From the plots above we conclude that:

1. The final system seems to overfit the training set (Chart 13), as also detected during the feature selection experiments. Nevertheless, we observe that the on-test curve clearly increases when new training data are added. We note that in the experiment of Chart 13, the test set (42 held-out user datasets) size is about 60% of the total training set size. As a result, for training data sizes below 60%, the system has been trained on fewer instances than it is being tested on. This contrasts with all learning curves we have presented so far, which used a fixed, much smaller test, at 10% of the total training set's size.
2. System performance in the tweet ranking use case (Chart 14) seems to be acceptable. This is even more stressed at high recall levels (Chart 15). This means that a user would need to scroll through few negative instances in the ranked list, in order to see all the “retweetable” tweets (or, in recall terms, to achieve recall=1.0).
3. Suppose a random baseline tweet ranker, with no background knowledge on the task and with no need of training, which always returns a random ranking of tweets. Using the 42 fresh, unseen user datasets, we evaluate this baseline in terms of Mean Interpolated Precision in Chart 16. When compared to the respective plot of our system (Chart 14), it turns out that our system achieves a much better performance than the random baseline. The baseline is never trained on data, therefore its performance is not correlated with the training dataset size.

5. Conclusions and future work

In this thesis we developed systems for personalised tweet filtering, in the form of retweetability prediction systems. This is a promising approach that has been followed by other researchers in the past, because it allows large numbers of tweets with their correct classes (retweeted or not retweeted) to be obtained through the Twitter API. We followed a single-model approach, where a global trained model offers predictions to all users, but the existence of user-aware features makes these predictions personalised. This way, it is possible to train the model with very large datasets and directly use it with new users, alleviating the cold start problem. After constructing a large corpus of Twitter data from 122 users (journalists), we examined how factors such as the volume of training data and the ratio of positive examples affect the prediction quality of our system. Then, we tested various setups, using Logistic Regression, J48 decision trees and various feature sets. Our final system, evaluated on fresh data, achieves F1 score equal to 0.473 (for the positive class), when the top 20 features are used. The comparable system of Uysal and Bruce Croft yielded 0.724 F1 score when all their features were used. On the other hand, when evaluated for its ability to rank tweets, our system exhibits very mean high interpolated precision at most recall levels.

Future work could consider using word embeddings for text representation, instead of the Bag-of-Words model. Our system was built with such provisions to allow a relatively easy text representation switch to word embeddings. Alternatively, LDA topic detection, combined with distribution similarities could be used to compare tweets, instead of vector cosine similarities. The feature set must also be studied more thoroughly, since we did not have enough time to test more combinations and to perform a qualitative analysis of the features used in the final system setup. As mentioned in Section 4.6, there are still open questions concerning the selected features and we consider their investigation very important. Furthermore, novel approaches, such as convolutional neural networks (CNNs) could be used as prediction models, since they have been found to offer impressive performance improvements in a wide range of tasks. Finally, the system could be tested with different dataset, such as the Twitter data of our previous work [Vougioukas, 2014], as well as in a real-life study with Twitter users. Such a study would be interesting to perform with both journalist and non-journalist users, allowing us to understand whether and under which conditions a system trained on a specific user type is extensible to other user types.

References

Alonso O., Marshall C. C., Najork M., "Are some tweets more interesting than others?#hardquestion", In Proceedings of the ACM Symposium on Human-Computer Interaction and Information Retrieval, Vancouver, Canada, pp. 2, 2013.

Androutsopoulos I., Language Technology course [PDF slides], Department of Informatics, Athens University of Economics and Business, 2015.

Berkovsky S., Freyne J., “Personalised network activity feeds: Finding needles in the haystacks”, Mining, Modeling and Recommending Things in Social Media, Springer International Publishing, pp. 21-34, 2015.

Chen J., Nairn R., Nelson L., Bernstein M., Chi E., "Short and tweet: experiments on recommending content from information streams", In Proceedings of the ACM Conference on Human Factors in Computing Systems, Atlanta, USA, pp. 1185-1194, 2010.

Chen K., Chen T., Zheng G., Jin O., Yao E., Yu Y., “Collaborative personalized tweet recommendation”, In Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval, Portland, USA, pp. 661-670, 2012.

Clinchant S., Perronin F., “Textual similarity with a Bag-of-Embedded-Words model”, In Proceedings of the ACM Conference on the Theory of Information Retrieval, Copenhagen, Denmark, p. 25, 2013.

Counts S., Fisher K., “Taking it all in? Visual attention in microblog consumption”, In Proceedings of the AAAI International Conference on Weblogs and Social Media, Barcelona, Spain, pp. 97-104, 2011.

Dahimene R., du Mouza C., “Filtering structures for microblogging content”, International Journal of Intelligent Information Technologies, Vol. 11, Issue 1, pp. 30-51, 2015.

Duan Y., Jiang L., Qin T., Zhou M., Shum H.Y., "An empirical study on learning to rank of tweets", In Proceedings of the ICCL International Conference on Computational Linguistics, Beijing, China, pp. 295-303, 2010.

Feng W., Wang J., “Retweet or not? Personalized tweet re-ranking”, In Proceedings of the ACM International Conference on Web search and data mining, Rome, Italy, pp. 577-586, 2013.

Garg A., Battiti R., Cascella R. G., " ”May I borrow your filter?” Exchanging filters to combat spam in a community", In Proceedings of the IEEE International Conference on Advanced Information Networking and Applications, Vienna, Austria, pp. 5 (Vol. 2), 2010.

Georgiou M., “Relevant snippet extraction in biomedical question answering”, M.Sc. thesis, Department of Informatics, Athens University of Economics and Business, 2015.

Hall M. A., “Correlation-based Feature Subset Selection for Machine Learning”, Ph.D. Thesis, University of Waikato, 1998.

Hall M., Frank E., Holmes G., Pfahringer B., Reutemann P., Witten I. H., “The WEKA data mining software: an update”, ACM SIGKDD Explorations Newsletter, Vol. 11, Issue 1, pp. 10-18, 2009.

Hosmer Jr D. W., Lemeshow S., “Applied logistic regression”, John Wiley & Sons, 2004.

Hurlock J., Wilson M. L., "Searching Twitter: Separating the Tweet from the Chaff", In Proceedings of the AAAI International Conference on Weblogs and Social Media, Barcelona, Spain, pp. 161-168, 2011.

Jenders M., Kasneci G., Naumann F., “Analyzing and Predicting Viral Tweets”, In Proceedings of the IW3C2 International World Wide Web Conference, Rio de Janeiro, Brazil, pp. 657-664, 2013.

Kapanipathi P., Jain P., Venkataramani C., Sheth A., “User Interests Identification on Twitter Using a Hierarchical Knowledge Base”, In Proceedings of the European Semantic Web Conference, Crete, Greece, pp. 99-113, 2014.

Khater S., Elmongui H., Gracanin D., “Tweets you like: personalized tweets recommendation based on dynamic users interests”, In Proceedings of the ASE International Conference on Social Informatics, Cambridge, USA, 2014.

Le Cessie S., Van Houwelingen J. C., “Ridge estimators in logistic regression”, Applied Statistics, Vol. 41, Issue 1, pp. 191-201, 1992.

Li B., Liu Y., “A novel approach for microblog message ranking based on trust model and content similarity”, International Journal of Database Theory and Application, Vol. 8, No. 3, pp. 289-296, 2015.

Manning C. D., Raghavan P., Schutze H., “Introduction to Information Retrieval”, Cambridge University Press, 2008.

Meier F., Elsweiler D., Wilson L. M., “More than liking and bookmarking? Towards understanding Twitter favouriting behaviour”, In Proceedings of the AAAI International Conference on Weblogs and Social Media, Ann Arbor, USA, 2014.

Metsis V., Androutsopoulos I., Paliouras G., "Spam filtering with naive bayes-which naive bayes?", In Proceedings of the Conference on Email and Anti-Spam, Mountain View, USA, pp. 27-28, 2006.

Mikolov T., Chen K., Corrado G., Dean J., “Efficient estimation of word representations in vector space”, arXiv preprint, arXiv:1301.3781, 2013.

Mikolov T., Sutskever I., Chen K., Corrado G., Dean J., “Distributed representations of words and phrases and their compositionality”, Advances in neural information processing systems, pp. 3111-3119, 2013.

Owoputi O., O'Connor B., Dyer C., Gimpei K., Schneider N., Smith N., “Improved Part-of-Speech tagging for online conversational text with word clusters”, In Proceedings of the NAACL Human Language Technologies: The Annual Conference, Atlanta, USA, 2013.

Pennington J., Socher R., Manning C., “Glove: Global Vectors for Word Representation”, In Proceedings of the Conference on Empirical Methods in Natural Language Processing, Doha, Qatar, Vol. 14, pp. 1532-1543, 2014.

Quinlan J. R., “C4.5: programs for machine learning”, Morgan Kaufmann, 1993.

Quinlan J. R., “Induction of decision trees”, Machine Learning, Vol. 1, Issue 1, pp. 81-106, 1986.

Rokach L., “Data mining with decision trees: theory and applications”, World Scientific, 2008.

Servia-Rodriguez S., Huberman B., “Deciding what to display: maximizing the information value of social media”, arXiv preprint, arXiv:1411.3214, 2014.

Tan C., Lee L., Pang B., “The effect of wording on message propagation: Topic-and-author-controlled natural experiments on Twitter”, arXiv preprint, arXiv:1405.1438, 2014.

Uysal I., Croft W. B., "User oriented tweet ranking: a filtering approach to microblogs", In Proceedings of the ACM International Conference on Information and Knowledge Management, Glasgow, Scotland, pp. 2261-2264, 2011.

Vougioukas M., “Development of a system to filter tweets”, B.Sc. thesis, Department of Informatics, Athens University of Economics and Business, 2014.

Waldner W., Vassileva J., “Emphasize, don't filter! Displaying recommendations in Twitter timelines”, In Proceedings of the ACM Conference on Recommender Systems, Foster City, USA, pp. 313-316, 2014.

Webberley W., Allen M. S., Whitaker M. R., “Inferring the interesting tweets in your network”, In Proceedings of the IEEE International Conference on Cloud and Green Computing, Karlsruhe, Germany, pp. 575-580, 2013.

Wu W., Zhang B., Ostendorf M., "Automatic generation of personalised annotation tags for twitter users", In Proceedings of the NAACL Human Language Technologies: The Annual Conference, Los Angeles, USA, pp. 689-692, 2010.

Zamani K., Paliouras G., Vogiatzis D., “Similarity-Based User Identification Across Social Networks”, Lecture Notes in Computer Science, Springer International Publishing, Vol. 9370, pp. 171-185, 2015.