ISTI@TREC Microblog track 2012: real-time filtering through supervised learning

Giacomo Berardi, Andrea Esuli, Diego Marcheggiani
Istituto di Scienza e Tecnologie dell'Informazione
Consiglio Nazionale delle Ricerche
56124 Pisa, Italy
firstname.lastname@isti.cnr.it

Abstract

Our approach to the microblog filtering task is based on learning a relevance classifier from an initial training set of relevant and non relevant tweets, generated by using a simple retrieval method. The classifier is then retrained using the (simulated) user feedback collected during the training process, in order to improve its accuracy as the filtering process goes on. In the official runs the system scored low effectiveness values, suffering a strong imbalance toward recall.

1 Introduction

Microblogging is a form of personal content sharing that derives from blogging, and it has a focus on concision, space and time locality, and social network-style interactions [3]. This is the second year that microblogging is the subject of a TREC track. For this edition two tasks have been proposed to the participants: Real-time Adhoc search (equivalent to the task in the 2011 track) and Real-time Filtering[1]. Twitter[2] is currently the dominant platform for microblogging, and it has been selected as the source of data for the microblog track.

Twitter enforces a strong policy on short messages, allowing only a maximum length of 140 characters[3]. In addition to simple text, a tweet can contain three types of references:

- URLs to external content. External links are typically used to extend a tweet with other media, e.g., an image shot by the user with her mobile phone, or to point to a web content the tweet comments upon.

[1] https://sites.google.com/site/microblogtrack/2012-guidelines

[2] http://twitter.com

[3] Twitter text length limitation derives from the original possibility of sending/receiving tweets on cell phones via SMS, which have a maximum length of 160 characters, with 20 characters reserved by Twitter for the id of the author of the tweet.

- Mention of names of other Twitter users, e.g., "*@aesuli*". This is a way to credit the mentioned user for the content of the tweet or a way to establish an exchange of public messages other user may be interested to read and join the discussion.

- *hashtags*: a hashtag is defined by any string prefixed with a #, e.g., "*#whyalwaysme*", "*#iwould*". The string can be a single word, an acronym, or multiple words joined together, and usually identifies the subject topic of the tweet (e.g., "*#SPIRE2011*") or expresses a comment about it (e.g., "*#epicwin*").

In this edition of the track we have participated to the Real-time Filtering task. The methodology for this task is the same of the *adaptive filtering* described in [7]. In the case of tweets, it simulates a scenario in which the system receives feedback from the user, about the relevance of tweets to a topic, so it can recalibrate the filter and improve the results. For each topic a time range is defined, the tweets are checked sequentially, starting from the oldest, in chronological order. No information is given to the system, unless it decides to classifies as relevant a tweet, then it can know the true relevance of it. No training data is available to the system, relevance judgment of a tweet can be accessed only in the filtering phase, after the system has added it to the results.

In our participation we have explored the use of a classification system in which a supervised learning model is retrained with a new example (tweet), each time this is classified as relevant to the filtering topic. In Section 2 we describe the Tweets2011 corpus; we have reused the data which was already downloaded for the 2011 microblog track. In Section 3 we describe the system developed for the real-time filtering task; we have combined a retrieval module, similar to the one used in the 2011 microblog track, and a supervised learning module. In Section 4 we show the evaluations of two runs of the system.

2 The Tweets2011 collection

The microblog track of this year reuses the Tweets2011 Corpus [4] of the previous year. We have thus reused the data we downloaded last year using the downloader provided by the organizers. Table 1 reports some statistics on the full corpus and on the various subsets we have defined in order to build our indexes, which are described in Section 3.

We have indexed only the tweets in English, filtering out non-English tweets. We have used a language recognition system that we have implemented along the lines of [2], in order to recognize English tweets. The English set, the only part we have actually indexed, resulted to be roughly a quarter of the entire corpus. We have then identified two subsets of the English set: a Hashtag set that contains only English tweets that have hashtags in them, and a Link set that contains only English tweets with URLs in them. The two subsets show a similar ratio of tweets with at least one hashtag/URL with respect to tweets without hashtag/URL (1:5.6 for hashtags, 1:6.6 for URLs).

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

Table 1: Some statistics from the corpus and the subsets we have selected for indexing. The total number of tweets is divided in effective tweets (http status 200), retweets (http status 302) and null tweets (http status 301/403/404). The hashtags column indicates the number of unique hashtags.

3 The adaptive filtering system

Our system, named CipCipPy[4], is an indexing and retrieval system based on the Whoosh[5] IR library, written in Python [1]. In this edition of the Microblog track we have extended it with to filtering task. The source code of CipCipPy is available for download at `http://hlt.isti.cnr.it/cipcippy/`.

The adaptive filtering task requires to decide, under real time processing constraints, if a tweet is relevant for a given query. We tackled it as a binary classification task, exploring the use of a supervised learning classification approach to the problem. We used a Naive Bayes classifier in order to have a quick training and classification time.

The Naive Bayes classifier requires a training set to learn on, composed by tweets that are examples of relevant or non-relevant content with respect to the topic. Given that at the beginning of the retrieval process there are no examples available, our system initially filters relevant tweets, while bootstrapping a training set, using the adhoc retrieval module of CipCipPy [1] (see Section 3.1).

Once the training set reaches a reasonable size (the size is a parameter of the system) that allows to train the classifier the bootstrap phase ends. The Naive Bayes classifier is then trained and it is used in the classification phase. During this phase the classifier is used to filter the tweet stream, and it is retrained any time a filtered tweet is considered to be relevant, adding such tweet to the training set with the proper relevance label assigned according to the true relevance judgment.

3.1 Ad Hoc retrieval module

We have built indexes of the English set for the first retrieval phase. There is an index specific to each topic, i.e, containing only the tweets prior to oldest known relevant tweet for the topic (*pre-topic index*), according to the relevance judgements produced by the 2011 evaluation. These indexes, built in this way, do not contain future evidence (e.g. document frequencies computed on the tweets to be filtered in real-time). We have not

[4] "Cip Cip" is the Italian word for the sound of birds, while "Py" identifies the Python programming language.

[5] `https://bitbucket.org/mchaput/whoosh/wiki/Home`

indexed retweets, since the guidelines and discussions in the mailing list of the track stated that retweets would be considered not relevant by default.

We have performed a retrieval phase in order to obtain an initial training set of relevant and non relevant examples. We have used the simple IDF as the weighting function in order to compute the retrieval score of the tweets. Including the term frequency component into the weighting could be not well suited for the task, given the short and relatively compact distribution of the text lengths.

An initial set of p relevant examples for each topic have been retrieved searching on the index of the English set. The queries have been formulated, from the original topic title, as the disjunction of the conjunction of every possible pair of terms appearing in the topic title, plus each single term. The top-p retrieved tweets have been selected to be positive examples in the training set.

Non relevant examples have been retrieved from the same index. In order to obtain a varied set of tweets that are non relevant to the topic, we have issued a query that is the negation of the disjunction of all the terms in the topic title. The top-n tweets in the ranking have been selected as negative examples for the initial training set.

Although the fact of searching for positive training examples in a set of tweets that by definition are likely to not contain relevant tweet may seem counterintuitive, note that we leave the determination of the p and n values to a set of parameter validation experiments on the validation topics (see Section 4).

During the initial phase of the filtering process, the query used to retrieve the positive examples from the pre-topic indexes is used to filter tweet, until the classification module is ready to be used.

3.2 Machine learning based filtering

The ad hoc retrieval module works on a part of the tweet corpus that is antecedent to the time span that is relevant to the filtering process. Only its query formulation contributes directly to the filtering process. The actual filtering system is composed by two modules, the bootstrap module and the classification module.

Bootstrap module The bootstrap module carries on the initial part of filtering process while the classifier is not yet trained, at the same time it contributes, along with the ad hoc retrieval module, to the definition of the initial training set for the classifiers. As described in the previous section, this module considers relevant any tweet that matches the query used to retrieve the positive examples from the pre-topic indexes.

Following the guidelines on the simulation of relevance feedback, for all the tweets considered relevant by the bootstrap module the relevance judgments are checked. They are then added to the training set, with their real relevance category. This process is repeated until b truly relevant tweets are found, i.e., until a minum number of positive examples are identified. The bootstrap module then stops and the filtering task is continued by the classification module, trained on the train set obtained by merging the training sets generated by the ad hoc retrieval and the bootstrap module.

Classification module The classification module classifies one tweet at the time, following their temporal order. When the classifier classifies a tweet as relevant, the real relevance is checked and the tweet is added to the training set with its true relevance class. After the addition the classifier is re-trained with the new training set, obtaining a new, updated classifier.

In our system we have used the Scikit-learn [5] implementation of the Multinomial Naive Bayes classifier. Obviously, the method can be instantiated with almost any learning algorithm, just taking into account the required balance between the potential accuracy of the resulting classifier and the cost, in time, necessary to train it.

3.3 Features

As features for the Naive Bayes classifier we have used the bag of words of the tweet status from the English set, plus some more tweet-specific features:

Linked page titles Words composing the title of the pages linked by a tweet (i.e., the text included in the `<title>` tag of the linked html Web page) have been added to the bag of words. The information contained in the linked page can contribute to the relevance of a tweet with respect to a query. For example, a tweet can be composed by a sentence expressing an opinion and a URL that links to a news, and the tweet becomes relevant because the content linked by the URL gives a proper, relevant, context to the opinion. On the opposite side, the whole content of a Web page can be misleading, due to the possibility of including unrelated information (e.g., navigational information, advertisement). For this reason we chose to include only the title of the Web page.

Hash tag splitting As we did in [1], in addition to the words of the tweet, we have used a hashtag splitter to split the compound words representing the hashtags in common English words. We have improved the Viterbi-based splitting model feeding it with a dataset larger than the one used in [1]. We have used the Google N-grams collection[6], taking the frequency of words from the English One Million collection of Google books from years 1999 to 2009.

Named entities We used a named entity recognizer specifically devised for tweets [6], to mark the presence in tweets of named entities of various types. In addition to the classic named entities i.e. "Person", "Location", "Organization", this system extracts other entities such as "Music Band", "Product", "Movie", "Sports Team" and "TV show". The rational behind this feature is that a tweet that does not contain the same type of named entities that (statistically) characterize the positive examples (i.e., the relevant tweets met during the filtering process until the actual time of the examined tweet) is likely a not relevant tweet for the topic under examination.

[6] http://books.google.com/ngrams/datasets

4 Results

For the official runs we have optimized the system parameters (p, n, b) by performing a validation phase using the topics indicated in the track guidelines (MB1, MB6, MB11, MB16, MB21, MB26, MB31, MB36, MB41, and MB46). The optimization has been based on the measure of F_1 (this choice was made before knowing the official evaluation measures, which included $F_{0.5}$ instead of F_1). We have explored all the combinations of values {0, 5, 10, 15, 20, 50, 100} for the three parameters. We have achieved the best validation scores for the configuration $p = 0$, $n = 100$, $b = 10$. This means that considering as positive examples even few relevant tweets retrieved prior to the oldest known relevant tweet has a negative impact on performance. Adding negative examples has instead a positive effect on the filtering process.

We have submitted two runs, one using external information (i.e., features from link titles and named entities), and one not using it. They are called *nemisExt* and *nemisNotExt* respectively. In Table 2 evaluations are showed, using the official evaluation measures; the runs are compared with best and median results, averaged on the topics. The obtained results are well below the median results of the track, except for recall. The system in fact resulted to be strongly imbalanced toward judging tweets as relevant. As expected, the use of external information improved the recall, though in this poor performace situation it is hard to determine the significance of the improvement.

	Precision	Recall	$F_{0.5}$	Utility
nemisExt	0.0293	0.4433	0.0343	0.0140
nemisNotExt	0.0315	0.4232	0.0370	0.0140
Median Average	0.1766	0.3343	0.1491	0.2076
Best Average	0.9224	0.9462	0.6073	0.5967

Table 2: Evaluations of the runs

The extreme imbalance between recall and precision is mainly generated by the query formulation we adopted in the bootstrapping phase. Moreover, making an a posteriori analysis of the system components we also identified a design error, dictated by efficiency goals in the choice of the learning algorithm. The Naive Bayes classifier uses prior knowledge on the probability of positive cases to take its classification decisions. By limiting, in the validation experiments, the value of n to a maximum value of 100, we have limited the prior probability of positive cases to a minimum value of $\min \frac{b+p}{n} = \frac{5}{100}$, when such probability is much lower (on average on topics, about $\frac{1}{113000}$). The fact that the $p = 10$ configuration has been selected instead of the $p = 5$ one is likely due to the fact that five (positive examples) is a really low absolute value, and doubling that value produces an improvement that easily counterweights any other negative effect related to prior probabilities. Although one can think about exploring the use of higher values for the n parameters, e.g., $n = 113000$, this is not a flexible solution with respect to possible fluctuations in the class prior distribution with respect to different topics and also requires a validation set that is representative of

the test set.

5 Future work

Our system did not perform well on the task, for two reasons: (i) the poor performance of the ad hoc retrieval system used in the bootstrapping phase, (ii) the choice of a supervised learning algorithm that relies on the prior probabilities from the training set. With respect to the first issue we plan to investigate the performance of the system when using one of the top-performing retrieval systems from the ad hoc retrieval task for the bootstrap phase. With respect to the second issue we plan to test the use other learning algorithms, such as SVMs, which do not use the prior probability of the class, in the classification phase.

References

[1] G. Berardi, A. Esuli, D. Marcheggiani, and F. Sebastiani. ISTI@ TREC Microblog track 2011: exploring the use of hashtag segmentation and text quality ranking. In *Proceedings of the Twentieth Text REtrieval Conference (TREC 2011)*, 2011.

[2] W. B. Cavnar and J. M. Trenkle. N-Gram-based text categorization. In *In Proceedings of SDAIR-94, 3rd Annual Symposium on Document Analysis and Information Retrieval*, pages 161–175, 1994.

[3] A. Java, X. Song, T. Finin, and B. Tseng. Why we twitter: understanding microblogging usage and communities. In *Proceedings of the 9th WebKDD and 1st SNA-KDD 2007 workshop on Web mining and social network analysis*, WebKDD/SNA-KDD '07, pages 56–65, New York, NY, USA, 2007. ACM.

[4] R. McCreadie, I. Soboroff, J. Lin, C. Macdonald, I. Ounis, and D. McCullough. On building a reusable twitter corpus. In *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '12, pages 1113–1114, New York, NY, USA, 2012. ACM.

[5] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine Learning in Python . *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[6] A. Ritter, S. Clark, and O. Etzioni. Named entity recognition in tweets: An experimental study. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1524–1534. ACL, 2011.

[7] I. Soboroff and S. Robertson. Building a filtering test collection for trec 2002. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, SIGIR '03, pages 243–250, New York, NY, USA, 2003. ACM.