Applying Opinion Mining to OSS selection process

PhD Minor Dissertation
Advisor: prof. Sandro Morasca

Davide Taibi[1]

Università degli Studi dell'Insubria
Via Carloni, 78, 22100 COMO, Italia

Abstract. Mining opinions in free text so as to separate the opinionated and the relevant content in text is a challenging research problem. People and organizations that are considering the adoption of Open-Source Software (OSS), or that need to choose among different OSS products are interested in knowing the user community's opinion, since this can provide useful indications about the strengths and limits of the software being evaluated. While several methods for the evaluation of the community size are available, there is no automated support to the extraction of the opinions of the community. In this paper we explore whether it is possible to automate the OSS selection process by means of automated Sentiment analysis techniques. Our goal is two-folded: on the one hand we want to improve the actual opinion mining techniques, on the other hand we want to apply this technique to catch user's opinions on OSS software. The first goal will be achieved by means a new Opinion Mining strategy (to be applied to the TREC'09 data set). The second goal will be pursued in two steps: first we shall develop a web crawler that is able to extract blogs posts on OSS, then we shall apply the opinion mining process the OSS blogs data-set.

1 Introduction

Sentiment Analysis and opinion mining are emergent research fields. Recently, with the born of several blogging platforms (see for instance Wordpress [1]) users express their opinions on every kind of topics, from politic to religion, from marketing to product reviews, etc.

This wealth of opinionated contents can be very useful to catch the user's thought and perception. We are interested in analyzing opinions of OSS because of the nature of its selection. The software selection process, and particularly OSS, often includes a preliminary phase in which the potential users collect information about the products by surfing in blogs, forums and newsgroups. Thus, the process of selecting OSS very often involves a long and boring web

[1] Wordpress - http://www.wordpress.com

search, looking into several forums, blogs and websites in order to extract as much information as possible.

In this work we describe the work done in the minor thesis. In Section2 we summarize our Opinion Mining approach, then, in Section 3 we show BlogAnalytics, a web crawler able to extract relevant blog posts from the web, and finally we describe the application of our Opinion Mining approach to a new data-set containing opinions on 32 OSS products. In Section 4 we draw conclusions and we outline future works while in Appendix5 we show the relevant techniques for selecting OSS, Opinion Mining and Sentiment analysis.

2 TREC'08 participation

In this section we summarize our proposal to TREC'08 [11].

We decided to apply to the Blog Track in TREC'08 in order to improve the existing Sentiment Analysis techniques. We participated both in opinion retrieval and blog distillation tasks. We aimed to test the effectiveness of learning methods in each of these tasks.

TREC'08 provided a very large data-set including a crawl of 100k blogs over an 11-week period. This data-set includes the blog postings (permalinks), feeds and homepages for each blog.

In our experiments we use only the permalinks component of the collection, which consists of approximately 3.2 million documents. For evaluating our methods we used the TREC Blogs06 test collection.

Detailed results are available in [11].

2.1 Opinion Retrieval

In the *Opinion Retrieval task*, first we indexed the collection, then we combined additional information, including the content of incoming hyperlinks and tag data from social bookmarking websites with our basic retrieval method (Divergence from Randomness version of BM25 (DFR_BM25) [1]).

The collection indexing phase was carried out by means of Terrier Information Retrieval system [20]. We extended this content-based retrieval technique with additional information including the content of incoming hyperlinks and tag data from social bookmarking websites. The latter has recently been shown to be useful for improving Web Search [14, 3, 25].

Our approach to ranking blog posts by their opinion level relies on a learning framework [27, 5]. We trained a Learning to Rank system to take both a relevance score (output by the rank learner described above) and an "opinion score" for each document into account when producing an output ranking. The advantage of this approach is that we do not need to explicitly decide how to combine these forms of evidence, but can rely on historical data for fine tuning the retrieval.

In order to identify the best retrieval model, we tested various state-of-the-art retrieval models including BM25 [21], Divergence from Randomness [2] and Language Modeling [28], for generating relevance scores for individual posts. We

Run	Mean Average Precision	Recall Precision	Recall Precision @10
DFR_BM25	0.2138	0.3836	0.4087
our baseline system	0.2663	0.2780	0.4273

Table 1. Topic-Relevance results for submitted baseline.

used the relevance assessments provided by TREC 2007 for the evaluation. The output of our analysis was that DFR_BM25 produced the best result, with Mean Average Precision (MAP) of 0.2138. Table 1 shows the topic relevance results of the DFR_BM25 method compared to our baseline system. The results indicate that using a rank learning approach to include additional information such as the content of incoming hyperlinks and tag data from social bookmarking websites, can boost the performance of a baseline content-based retrieval system.

The problem, then, is to estimate a score for the "opinionatedness" of each document. We have two approaches to doing this. In the first approach, we calculate an opinion score for each term in the document and then we combine the score over all terms in the document. In the second, we train a classification system to distinguish between opinionated and non-opinionated posts. Then, we use the confidence of the classifier as an opinion score for the document.

For a complete description of the model see [11].

2.2 Blog Distillation

Blog search users often wish to identify blogs about a given topic so that they can subscribe to them and read them on a regular basis [16]. The blog distillation task can be defined as: "Find me a blog with a principle, recurring interest in the topic X." Systems should suggest feeds that are principally devoted to the topic over the timespan of the feed, and would be recommended to a user as an interesting feed about the topic (i.e a user may be interested in adding it to his RSS reader).

The blog distillation task has been approached from many different points of view. In [9], the authors view the distillation task as an as ad-hoc search and they consider each blog as a long document created by concatenating all postings together. Other researchers treat it as the resource ranking problem in federated search [10]. They view the blog search problem as the task of ranking collections of blog posts rather than single documents. A similar approach has been used in [22], where they consider a blog as a collection of postings and use resource selection approaches. Their intuition is that finding relevant blogs is similar to finding relevant collections in a distributed search environment. In [16], the authors modelled blog distillation as an expert search problem and use a voting model for tackling it.

Our intuition is that each post in a blog provides evidence regarding the relevancy of that blog to a specific topic. Blogs with more (positive) evidence are more likely to be relevant. Moreover, each post has many different features like content, in-links, and anchor text that can be used to estimate relevancy.

There are also global features of each blog like the total number of posts, the number of postings that are relevant to the topic and the cohesiveness of the blog that could be useful to consider.

Our first approach is to create a baseline for blog distillation system that uses only the content of blog posts as a source of evidence. To do this, we consider the expert search idea proposed in [13]. The main idea of that work is to treat blogs as experts and feed distillation as expert search. In the expert search task, systems are asked to rank candidate experts with respect to their predicted expertise about a query, using documentary evidence of expertise found in the collection. So the idea is that the blog distillation task can be seen as a voting process: A blogger with an interest in a topic would send a post regularly about the topic, and these blog posts would be retrieved in response to the query. Each time a blog post is retrieved, it can be seen as a vote for that blog as being relevant to (an expert in) the topic area.

We use the voting model to find relevant blogs. The model ranks blogs by considering the sum of the exponential of the relevance scores of the postings associated with each blog. The model is one of the data fusion models which Macdonald and Ounis used in their expert search system[15].

For our second approach, we used more features to represent each blog beside its content. To take the different features into account, we use a Rank Learning approach [27] to combine the features into a single retrieval function. Useful features are:

– Cohesiveness of blog postings
– Number of postings
– Number of relevant postings (posts in top N relevance results)
– Number of inlinks
– Relevance of inlink post content
– Relevance of inlink anchor-text

2.3 TREC'08 Experimental Results

For our TREC 2008 participation we trained a Learning to Rank system to rank results by combining our relevance score for the document with Kullback-Leibler divergence [17] and Support Vector Machine (SVM) classifier as described in details in [11].

In order to calculate the opinion score for documents we used the expected opinionatedness of words in document as described before.

To find relevant posts for each topic, the Divergence from Randomness version of BM25 (DFR_BM25) weighting model was used to compute a score for each blog post [20]. We chose this weighting model based on its performance on TREC'06 and '07 blog post opinion retrieval baselines. For each query we selected the 20,000 highest scoring postings as $R(Q)$ and use the voting model to combine post relevance scores into blog relevance scores. Figure 1 shows the performance of our baseline for each query compared to best, worst and median precision for that query among all groups in TREC'08, where queries are

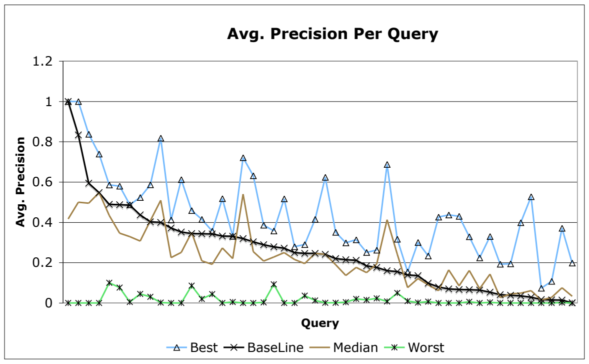

Fig. 1. Average Precision for each Query (ordered by relative performance) for the blog distillation task using a simple voting model. The best, median and worst scores are those of the other participants in TREC'08.

Feature Name	Description
Blog Relevancy	$\sum_{p \in R(Q) \cap B} exp(score(p, Q))$
Cohesiveness	$\sum_{p \in B} KL(p\|\|B) \div \|B\|$
Normalized Blog Relevancy	Blog Relevancy $\div \|B\|$
Fraction of Relevant Postings	$\|B \cap R(Q)\| \div \|B\|$
Inlink Content Relevancy	$\sum_{d \in inlinks\text{-}source(B)} exp(score(d, Q)) \div \|inlinks(B)\|$
Inlink Anchor-Text Relevancy	$\sum_{a \in inlinks\text{-}anchor\text{-}text(B)} exp(score(a, Q)) \div \|inlinks(B)\|$

Table 2. Selected features for learning phase of blog distillation approach.

sorted based on our baseline precision[2]. We saw that our baseline has reasonable performance across the queries and outperforms the median for most of the queries.

For the second experiment we extracted the features in Table 2 and we used a rank learning system (SVM-map[3]) to learn a ranking function over the features [27]. For the learning phase we trained the rank learner on the 45 topics and their corresponding relevance assessments used in the TREC'07 blog distillation task. We then tested the model on the 50 topics used for TREC'08.

Figure2 shows the precision-recall curve for the trained ranking model compared to the baseline expert search approach. We see that using other features beside content doesn't help in retrieving better blogs. These results are consistent

[2] Jonathan Elsas suggested this representation in his weblog (http://windowoffice.tumblr.com/)

[3] (http://projects.yisongyue.com/svmmap/

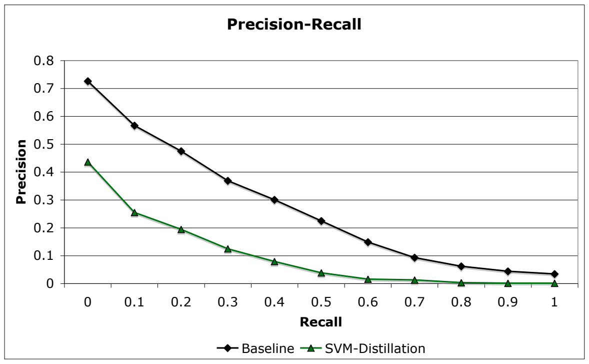

Fig. 2. Precision-Recall for Baseline and Rank learning model

with the results reported in [13], where the use of anchor text and cohesiveness didn't appear to improve performance. The reason is possibly due to the data set itself. Only 3% of postings have a link to another posting inside the data set and even in this small percent of links there are some noisy and non-informative links. Thus we will need to look for a better way to use this information. Furthermore, the small amount of training data available - only 45 queries - may have affected the ability of the rank learner to learn a model that well generalized to the test data.

Although we couldn't improve the baselines, the comparison between results obtained through Kullback-Leibler divergence and SVM, shows that using the SVM confidence as an opinion score works better than the expected opinionatedness approach on our baseline, but the reverse is true for the TREC baseline.

After the TREC submissions, we repeated the experiments performing a more comprehensive study and using 10-fold cross-validation. We discovered that even for the TREC baselines, using an SVM to calculate opinion scores for documents works better than the expected opinionatedness approach. The results showed an improvement of about 17% over TREC baseline.

A distinct advantage of our "multiple levels of learning" approach is that we maximize the use of available training data and thus do not need to rely on external sources of opinion-bearing word lists, that may not be well-suited to the blog opinion retrieval task.

3 Applying TREC'08 model to OSS reviews

In this section, first we describe BlogAnalytics, then we apply the Opinion Mining approach described in Section 2, on 32 different data-sets containing posts about a set of OSS products.

3.1 OSS data-set building with BlogAnalytics

In order to build a data-set containing posts that express an opinion about a particular topic, we developed BlogAnalytics[4]. BlogAnalytics queries blog search engines (like Technorati [5] and Google log Search [6]) every day, extracting the list of relevant posts for a given topic.

The complete data-set, and BlogAnalytics source code are freely available under GPL license on BlogAnalytics web-site. Each data-set includes an xml index summarizing each post, a folder containing all html pages, and a database dump with some meta-information. BlogAnalytics stores all post information on a relational database to speed up the post selection up by means of SQL queries. Furthermore, BlogAnalytics extracts pages by using the same format of TREC, so the application of techniques developed in TREC can be quickly applied to the generated data-sets. Further details on BlogAnalytics data, software architecture and documentation can be downloaded on the project website.

Once we developed BlogAnalytics, we selected a set of projects on which extract posts.

The selection of projects addressed different types of software applications, generally considered stable and mature. The complete set of projects comprises 32 products, having different characteristics:

– There are different types of programs and applications (from web servers to operating system, from libraries to content management systems).
– The communities of developers and users have different sizes.
– The projects have different ages.

During the selection it was taken care that every factor value was present in the set of projects. BlogAnalytics ran for 4 months, from December 1, 2008 to March 9, 2009, collecting thousands of posts for each OSS project. In Table 3 we show the projects list and the number of posts extracted. In Fig. 3 we show the trend of the number of posts retrieved in the same period. Considering that the dataset is automatically built, it does not include any training set neither validation set.

In Figure 3 we can see an interesting trend for each project. What we can clearly see is that there are no major intersections and all projects follow a similar trend. We reserve the possibility of more investigation in future works.

3.2 Opinion mining task application

In this section we want to apply the same techniques, applied in Section 2, for each OSS data-sets:

1. We indexed the collection retrieving the most relevant posts
2. We used SVM to give opinion score to terms included in the collection

[4] BlogAnalytics, http://www.BlogAnalytics.it
[5] Technorati, http://www.technorati.com
[6] Google Blog search, http://blogs.google.com

OSS Project	Number of Posts
Apache httpd server	3550
BusyBox	1169
CVS Concurrent Versions System	3564
DDD data display debugger	3598
DotNetNuke	1866
Drupal	8904
eclipse TPTP	2695
eZ publish	2612
GNU Project Debugger	2784
jasper report	2674
JBoss	3557
JMeter	524
Joomla	9477
linux debian	2305
Mambo CMS	2256
MySQL	19874
OpenSSL	1995
PHPNuke	816
Plone	1289
postgresql	2916
Servicemix	335
Subversion SVN	1730
Talend	575
Weka	553
Xerces	778
Xoops	2261
ZFS	3874

Table 3. Number of posts per each project from December 1, 2008 to March 9, 2009.

3. We used top 1000 ranked terms as features
4. In order to assign opinion score to documents, we trained a SVM based on relevance judgements that we obtained in step 1. Training data are represented based on 1000 selected features.
5. We used the classifier to classify test document. The confidence of the classifier was then used as opinion score for documents.
6. We combine relevance and opinion score(from step 4) by means of SVM and we produce the final ranking.

The collection indexing was carried out by means of Terrier Information Retrieval system [20]. Like in Section 2, we used the Divergence from Randomness version of BM25 (DFR_BM25) weighting model to compute a score for each blog post.

Then, we extracted the features in Table 2. We used SVM-map as a rank learning system to learn a ranking function over them [27]. Considering that we don't have a relevance assessment and neither a opinion assessment on the OSS data-sets, for the learning phase we trained the rank learner on the 45 topics and

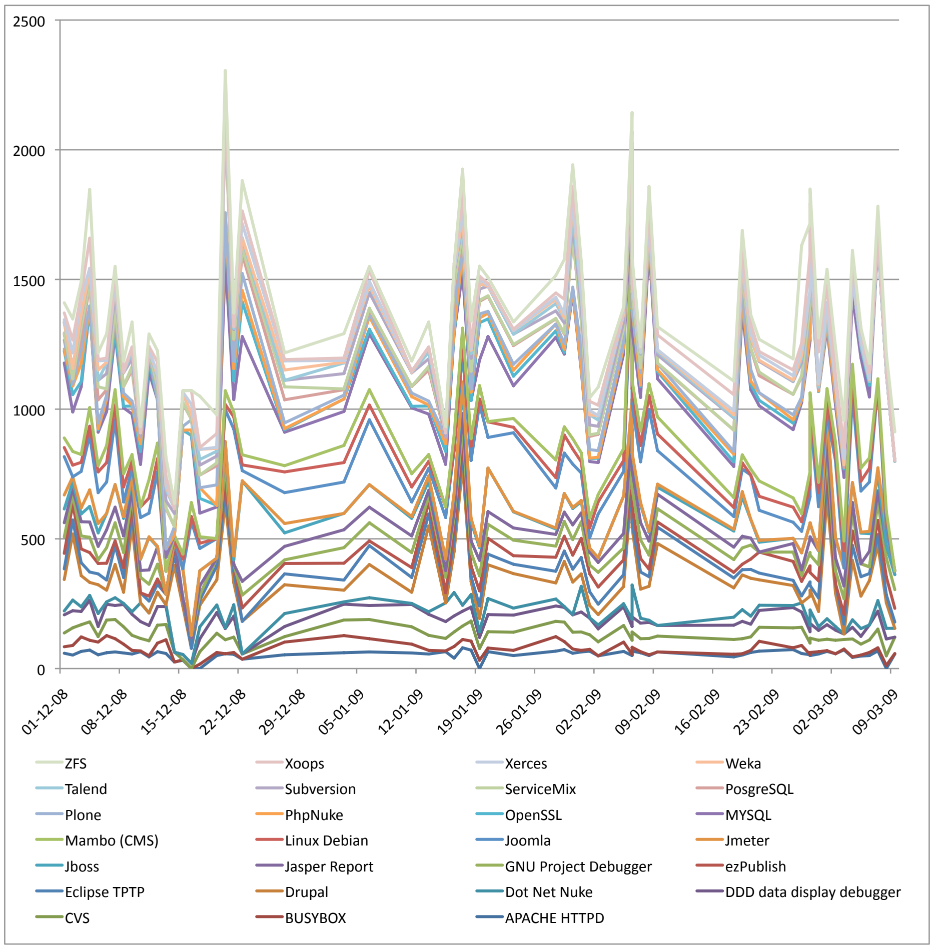

Fig. 3. Posts trend per week per project.

their corresponding relevance assessments used in the TREC'07 blog distillation task.

Considering that we don't have a training data-set on OSS products, the learning phase was carried out by training the rank learner on the 45 topics and their corresponding relevance assessment used in TREC'08 blog distillation task.

Finally we tested the model on the OSS data-set.

Taking into account that we don't have any validation set, we were not able to provide a precision-recall curve for the trained ranking model. We also think that a disadvantage of our "multiple levels of learning" approach is that we can not maximize the use of training data because of its unavailability. We manually checked for some results and we discover that most of results were wrong. Considering that the validation has been done manually, and just for a small sample of post, we are not able to determinate the accuracy but we can just say that results are not reliable to proceed with the sentiment analysis step.

The reason is possibly due to the data set itself. The TREC'08 data-set is composed by an heterogeneous set of topics, ranging from politics to science, from news to art. We used TREC'08 data-set as training data, and most of features included in TREC'08 data-set are completely out of scope for OSS projects. Our data-set is domain specific and need an ad-hoc training set to be set-up.

This work was extremely useful to test the existing opinion mining and to learn the Sentiment Analysis background.

In the next future, we are planning to manually create a validation set in order to test the real effectiveness of the method for OSS products.

4 Conclusions

In this work we defined an approach for analyzing opinions of OSS users expressed on blogs and forums. First we improved the actual Opinion Mining techniques. Then, we created Blognalytics, a tool able to build up our own data-set. We run BlogAnalytics for four months extracting the relevant blog posts for a set of 32 well known OSS projects. The data-set contains thousands of posts for each OSS project and is freely available (under GPL license) on BlogAnalytics web-site.

Taking into account our TREC'08 application, in the Opinion Retrieval task we tried a learning approach for generating opinion scores for individual documents and also for learning a ranking function that combines opinion evidence with relevancy into a single ranking function. A distinct advantage of our approach is that by performing multiple levels of learning, we maximize the use of available training data while not relying on external sources of opinion-bearing word lists, that may actually not be well-suited to the blog opinion retrieval task. In the Distillation task we have implemented a baseline that has reasonable results compared with other systems. However we have learned that blog distillation performance does not seem to improve when using the information contained in links between posts in a blog data set. We conjecture that this is because some links are not useful and we should find a way to use this link information only when probability of its informativeness is high. To do this, we plan to focus only on links to blogs that are related to the topic and analyze their structure. Since the most important blogs about a topic (those that contain the most information) will likely also be the most influential, other related blogs would link to them.

Considering the application of our approach to the set of OSS projects, we find out that, at the moment, the Opinion Retrieval step, based on TREC'08 training data is not directly applicable to the OSS projects review. In order to rely on the Opinion Mining framework, in the next future we will analyze all data-sets by hand and we will create training data that will be freely available.

Once we built the training data, we will be able to create an automated tool to collect and automatically analyze opinions for OSS projects.

5 Acknowledgments

The research presented in this paper has been partially funded by the IST project "QualiPSo", sponsored by the EU in the 6th FP (IST-034763); We are deeply indebted to and would like to thank University of Lugano, in quality of Prof. Fabio Crestani, Shima Gerani and Mark Carman for the collaboration in this work.

References

1. G. Amati, C. Carpineto, and G. Romano. Fondazione ugo bordoni at trec 2004. In *Proceedings of the Thirteenth Text REtrieval Conference (TREC 2004)*, 2004.
2. Gianni Amati and Cornelis Joost Van Rijsbergen. Probabilistic models of information retrieval based on measuring the divergence from randomness. *ACM Transactions on Information Systems (TOIS).*, 20(4):357–389, 2002.
3. Shenghua Bao, Guirong Xue, Xiaoyuan Wu, Yong Yu, Ben Fei, and Zhong Su. Optimizing web search using social annotations. In *WWW '07: Proceedings of the 16th international conference on World Wide Web*, pages 501–510, 2007.
4. Eric Brill. Some advances in transformation-based part of speech tagging. In *National Conference on Artificial Intelligence*, pages 722–727, 1994.
5. Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. Learning to rank using gradient descent. In *ICML '05: Proceedings of the 22nd international conference on Machine learning*, pages 89–96. ACM, 2005.
6. Eugene Charniak. A maximum-entropy-inspired parser. In *Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference*, pages 132–139, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.
7. M. Chinosi, V. del Bianco, L. Lavazza, S. Morasca, and D. Taibi. How european software industry perceives oss trustworthiness and what are the specific criteria to establish trust in oss. Web published. Accessed May 2009, 2008.
8. F.W. Duijnhouwer and C. Widdows. Open source maturity model. Web published. Accessed Janaury 2009, 2003.
9. M. Efron, D. Turnbull, and C. Ovalle. University of Texas School of Information at TREC 2007. In *Proc. of the 2007 Text Retrieval Conf*, 2007.
10. Jonathan L. Elsas, Jaime Arguello, Jamie Callan, and Jaime G. Carbonell. Retrieval and feedback models for blog feed search. In *SIGIR*, pages 347–354, 2008.
11. S. Gerani, M. Keikha, M. Carman, R. Gwadera, D. Taibi, and F. Crestani. University of lugano at trec 2008 blog track. In Ellen M. Voorhees and Lori P. Buckland, editors, *TREC*, volume Special Publication 500-277. National Institute of Standards and Technology (NIST), 2008.
12. Daniel Gruhl, R. Guha, Ravi Kumar, Jasmine Novak, and Andrew Tomkins. The predictive power of online chatter. In *KDD '05: Proceeding of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 78–87. ACM Press, 2005.

13. D. Hannah, C. Macdonald, J. Peng, B. He, and I. Ounis. University of Glasgow at TREC 2007: Experiments in Blog and Enterprise Tracks with Terrier. In *Proceedings of TREC*, 2007.
14. Paul Heymann, Georgia Koutrika, and Hector Garcia-Molina. Can social bookmarks improve web search? In *WSDM '08: Proceedings of the First ACM International Conference on Web Search and Data Mining*, 2008.
15. C. Macdonald and I. Ounis. Voting for candidates: adapting data fusion techniques for an expert search task. In *Proceedings of the 15th ACM international conference on Information and knowledge management*, pages 387–396. ACM Press New York, NY, USA, 2006.
16. C. Macdonald, I. Ounis, and I. Soboroff. Overview of the trec-2007 blog track. In *Proceedings of the Sixteenth Text REtrieval Conference (TREC 2007)*, 2007.
17. Christopher D. Manning and Hinrich Schtze. *Foundations of Statistical Natural Language Processing.* The MIT Press, June 1999.
18. Oliver Mason. Qtag - a portable probabilistic tagger, 1997.
19. G. Mishne and N. Glance. Predicting movie sales from blogger sentiment. In *AAAI 2006 Spring Symposium on Computational Approaches to Analysing Weblogs*, 2006.
20. I. Ounis, G. Amati, V. Plachouras, B. He, C. Macdonald, and C. Lioma. Terrier: A High Performance and Scalable Information Retrieval Platform. In *Proceedings of ACM SIGIR'06 Workshop on Open Source Information Retrieval (OSIR 2006)*, 2006.
21. Stephen Robertson, Hugo Zaragoza, and Michael Taylor. Simple bm25 extension to multiple weighted fields. In *CIKM '04: Proceedings of the thirteenth ACM international conference on Information and knowledge management*, pages 42–49, 2004.
22. J. Seo and W.B. Croft. UMass at TREC 2007 Blog Distillation Task. In *Proc. of the 2007 Text Retrieval Conf*, 2007.
23. D. Taibi, L. Lavazza, and S. Morasca. Openbqr: a framework for the assessment of oss. *International Journal on Open Source Development, Adoption and Innovation*, pages 173–186, 2007.
24. Peter Turney. Thumbs up or thumbs down? semantic orientation applied to unsupervised classification of reviews.
25. Yusuke Yanbe, Adam Jatowt, Satoshi Nakamura, and Katsumi Tanaka. Can social bookmarking enhance search in the web? In *JCDL '07: Proceedings of the 2007 conference on Digital libraries*, pages 107–116. ACM, 2007.
26. K. Yang, N. Yu, and H. Zhang. Widit in trec 2007 blog track: Combining lexicon-based methods to detect opinionated blogs. In *Proceedings of the Sixteenth Text REtrieval Conference (TREC 2007)*, 2007.
27. Yisong Yue, Thomas Finley, Filip Radlinski, and Thorsten Joachims. A support vector method for optimizing average precision. In *SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 271–278, New York, NY, USA, 2007. ACM.
28. Chengxiang Zhai and John Lafferty. A study of smoothing methods for language models applied to information retrieval. *ACM Transactions on Information Systems (TOIS)*, 22(2):179–214, 2004.
29. Q. Zhang, B. Wang, L. Wu, and X. Huang. Fdu at trec 2007: Opinion retrieval of blog track. In *Proceedings of the Sixteenth Text REtrieval Conference (TREC 2007)*, 2007.
30. Wei Zhang, Clement Yu, and Weiyi Meng. Opinion retrieval from blogs. In *CIKM '07: Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pages 831–840, New York, NY, USA, 2007. ACM.

A Relevant Techniques

In this section first we show Information Retrieval context that each year helps researchers in improving their algorithms describe then we listed some of the most important Opinion Mining and Sentiment Analysis (aka Opinion Mining) techniques and finally we show why opinions are important in selecting OSS.

A.1 Information Retrieval Contexts

Since 1992, the born of challenges (see for instance trec[7] and SIGIR [8]) encourage research within the information retrieval community by providing the infrastructure necessary for large-scale evaluation of text retrieval methodologies. For Example, each year, TREC provides a test set of documents and questions on which participants can run their own retrieval systems, and return a list of the retrieved top-ranked documents. Then, TREC collect and judges the retrieved documents for correctness, and evaluates the results. The TREC cycle ends with a workshop that is a forum for participants to share their experiences. TREC and other similar contexts help significantly the community in improve their Information Retrieval algorithms by means of the comparison of the results of several people on the same problem. TREC is composed by several tracks Each track is focused in a particular retrieval task acting as incubator for new research areas: during the first year of a track a problem is defined by creating the necessary infrastructure to support the research (test collections, evaluation methodology...). The opinion retrieval task was introduced in TREC 2006 and continued since 2008. In 2007, more than 20 groups participated to TREC by using a 3-step algorithm. The most important techniques are shown in Opinion Mining section. This year there were 7 tracks: Blog, Chemical IR, Entity, Legal, Million Query, Relevance Feedback and Web Track. We are interested in Blog track collection for our experiment . The collection contains 3.2 million documents collected in December 2005.

A.2 Opinion Mining and Sentiment Analysis Approaches

Recently, a good deal of work has been done by researchers on sentiment analysis (or polarity analysis) of reviews and opinion minings. Opinion mining is an important step where researchers try to understand if a text contains an opinion while Sentiment Analysis is a step further, involving polarity analysis detecting if an opinion is positive or negative.

The most important work in Opinion Mining comes from TREC. The vast majority of Opinion Mining techniques applied a 3-step algorithm. [30], first they decompose documents in sentences, then labelled each sentence with an opinion score by means of a classifier. Sequently they labelled the text if it contains at least one opinionated sentence. We refer the interested readers to

[7] TREC: Text REtrieval Conference (http://trec.nist.gov)
[8] Information Retrieval In Context - SIGIR (http://ir.dcs.gla.ac.uk/context/)

[30]. [26] considered multiple sources of evidence in their opinion retrieval steps by combining scores from opinion detection based on a common opinions terms. Then they built a lexicon by identifying the most occurring terms. Sequently, the opinion terms are manually labelled by assigning an opinion weight. They identify also the low frequency terms score, and acronyms score. Then, they combined the obtained each score as a weighted sum and they used this sum as training data. Finally in the third step, the linear combination of relevance and opinion score is used to score and rank documents. In another paper, Zhang et al. [29] used the Classification by Minimizing Errors (CME) to assign an opinion to each sentence of a blog post. They assigned an opinion score to each document by means of an SVN classifier, based on the values of the defined features. They used a set of movie review [9] to train the CME classifier and they used BLOG'06 collection for classifying documents by means of SVM. The blog posts were ranked by the final score that was calculated as the relevance score times the opinion score.

Taking into account Sentiment Analysis techniques, typically, the methods employed include combinations of machine learning and shallow natural language processing methods, and achieve good accuracy [19]. For instance, a recent study showed that peaks in references to books in weblogs are likely to be followed by peaks in their sales [12].

The year 2001 or so, seems to be the beginning of widespread awareness of the research problems and opportunities that sentiment analysis and opinion mining raise. Factors behind this widespreads include the rise of machine learning methods in natural language processing and information retrieval, the availability of data-set for machine learning algorithms to be trained. In 2002, Bo Pang and Lillian Lee[1], applied three Machine Learning techniques on the Movie review Domain: Support Vector Machines (SVMs), Naive Bayes and Maximum Entropy. They tested the algorithms on unigrams and bigrams, appending POS tags to every word via Oliver Mason's Qtag program [18]. This serves a crude form of word sense disambiguation distinguishing the different usage of words (e.g. the difference in usages of the word "love" in "I love this movie" versus "This is a love story". They looked at the performance of using adjectives alone, discovering that adjectives provided a less useful information then unigram presence. Indeed, simply using the most frequent unigrams presence information turned out to be the most effective way to spot opinions in text, yielding performance comparable to that using the presence of all lines. In terms of relative performance, Naive Bayes tend to the worst while Support Vector Machines tend to the best. An important problem come out from this paper was the needs of the identification of some kind of features indicating weather sentences are on topic.

In the same year, Peter Turney comes out with a different solution based on an unsupervised algorithm for classifying reviews on the web as "thumbs up" or "thumbs down" [24]. His solution is based on the semantic orientation (SO) of adjectives and verbs calculated as the mutual information between given phrases

[9] The movie review data-set is available on-line at http://www.cs.cornell.edu/people/pabo/ movie-review-data/

and the word ”excellent” minus the mutual information between the given phrase and the word ”poor”. A review is classified as recommended if the average semantic orientation of its phrases is positive. The algorithm was tested on four different domains (reviews of automobiles, banks, movies and travel destinations). This algorithm use a natural language processing technique (NLP) called *part-of-speach*. The part-of-speech of a word is a linguistic category that is defined by its syntactic or morphological behaviour. Common POS categories are: nouns, verbs, adjectives, pronouns, prepositions, conjunctions and interjections. POS tagging, by labeling (or tagging) each word in a sentence with its appropriate part-of-speach, is made by using the Penn Treebank POS Tags [6]. The approach is based on three steps: first a part-of-speech tagger (POS) is applied to the review [4], extracting phrases containing adjectives or adverbs. Then, the algorithm extract two consecutive words where one of them is an adjective/verb ad the ther is a context word. Two consecutive words are extracted if their tag are conform to any of pattern in 4. The JJ tags indicate adjectives, the NN tags are nouns, the RB tags are verbs. NNP and NNPS (singular and plural proper nouns) are avoided, so that names of the items in the reviews cannot influence the classification.

Table 4. Pattern of tags for extracting two-word phrases

First Word	Second Word
JJ	NN or NNS
RB, RBR, or RBS	JJ
JJ	JJ
NN or NNS	JJ
RB, RBR or RBS	VBN or VBG

In the second step the algorithm estimate the semantic orientation of the extracted phrases using the PMI-IR algorithm, measuring the strength of semantic association between two words by using mutual information. Where p(word1 & word2) is the probability that word1 and word2 co-occur is calculated as: SO(phrase)= PMI (phrase, ”excellent”) PMI (phrase, ”poor”). The probability is calculated by means of queries to a search engine and collecting the number of hits. Thus, by searching the two terms together and separately, it is possible to estimate the point-wise mutual information. In the third and last step, In the last step, given a review, the algorithm computes the average SO of all phrases in the review and classifies the reviews as recommended if the average SO is positive, not recommended otherwise.

A.3 How People select OSS

The definition of the information commonly used by the users when they evaluate OSS projects has been investigated in the last few years, and several OSS evaluation methods have been proposed (see for instance [23], [8]). In this context, the QualiPSo project [10] is an ongoing initiative that has been funded by the EU since 2006 with the goal of providing a coherent and systematic evaluation of OSS trustworthiness, based on the assessment of a common and widely adopted set of factors. One of the objectives of the QualiPSo project is to understand the reasons and motivations that lead software companies to adopt or reject OSS and to understand how people can trust software. In QualiPSo, we have identified [7] a common set of factors that users take into account when selecting OSS. These factors have been obtained via a number of interviews carried out in industrial contexts, to elicit the evaluation factors directly from industrial players. Users opinions, reliability of source code and short-term support have turned out to be some of the most important factors in our analysis [7] .

Our work differs from existing studies of sentiment analysis and business data in two important aspects. First, our domain includes weblogs from several sources i.e., a set of domains which tend to be far less focused and organized than the typical product review data targeted by sentiment analyzers, and consist predominantly of informal and unorganized text. Second, we aim at applying this study as a specific means to understand if the opinion of the OSS community on an OSS product reflect the software quality.

[10] http://www.qualipso.eu