Topical Evenlope of Tweets via Hashtag Classification

Mengying DU

Affiliation not available

February 8, 2016

1 Introduction

With over 300 million registered users producing over 500 million tweets per day[1], Twitter has became one of the largest and most popular microblogging websites. To cope with the volume of information shared daily, Twitter has introduced hashtags, keywords prefaced with "#", to help users categorize and search for tweets, which provide a way for a user to indicate the subject of a tweet in a way that is easy to search for as featured index. However, not all the hashtags are in a "good" format. For example, when a big event just happened, some of the hashtags were not represented in a uniform way or missed some keywords. To retrieve all the information related to this big event, it asks for classifying the tweets and reproducing relatively uniform hashtags. To made this task reachable, I made a simple assumption that a top trends tweet's hashtag content is a good approximation of its total content[7].

In this paper I present an algorithm to learn the relationships between the literal content of a tweet and the types of hashtags that could accurately describe that content. To better classify each tweet, I need to overcome problems of text cleaning, dimensionality reduction, and multi-class categorization.

2 Related Work

Currently, Twitter has not implemented any hashtag classification system which suggests and predicts appropriate hashtags for the users' tweets. In the research literature, there are works related to hashtag clustering and hashtag prediction. In this section, I introduce several recent papers have addressed the classification on twitter hashtags network.

Godin et al.[4] propose a method for hashtag recommendation. It is a unsupervised system, in which the classification is based purely on the content. Their approach relies on Latent Dirichlet Allocation (LDA). This system makes use of a topic distribution. LDA is a hidden topic model. This model assumes that there is a limited set of T topics where a document (in this case a tweet) can be about. A topic distribution over these topics will be assigned to each tweet. In the end, the topic which has the highest value assigned to it, will be selected by the system as the valid topic for that tweet. After the topic is assigned to a tweet, five hashtags will be selected. The selection was done by using topic-term count values, determining the top words of every topic in ranked order. These top words were then converted to hashtags. Because the researchers considered evaluation a subjective task, two persons were used to independently evaluate the applicability of the suggested hashtags. In cases of disagreement, there was a discussion until agreement was reached. For the evaluation, 100 tweets were randomly picked from the original set, which contained 1.8M tweets. For 80 of the 100 tweets in the test set, at least one appropriate hashtag could be selected by the judges, out of a set of five hashtag suggestions. When using a set of ten hashtags instead of five, accuracy increased to 91 appropriate suggestions.

Zangerle et al.[8] compare three different approaches to recommend hashtags based on a tf-idf representation of the tweet.The tf-idf score for a given tweet is calculated by taking the sum of the tf-idf's of all

separate terms in the tweet. They rank the hashtags based on the overall popularity of the tweet, the more terms are matched between tweets, the higher the ultimate tf-idf score will be.

$$tfidf_{t,d} = tf_{t,d} \times idf_t \tag{1}$$

Three different approaches for the ranking of hashtags were tried. (1) Overall popularity of the hashtag on Twitter, (2) frequency counts of the specific hashtags in the similarity set and (3) similarity between the source tweet and the tweet in which the candidate hashtag appears. This last approach turned out to yield the most appropriate hashtag ranks. Poschko[6] firstly argues that two hashtags are similar if they co-occur in a tweet. He creates a clustered graph with co-occurrence frequency as the distance measure. Furthermore, Li[5] recommend hashtags from similar tweets by using WordNet similarity information and an Euclidean distance metric. Based on the found distances between words, the system predicts the hashtags. It collects a few of the most similar tweets, extracts their hashtags and then decides which hashtag to pick, based on tag ratios. When one tag gets a ratio higher than 50% it will be chosen as the predicted tag. The system ended up predicting around 86% of hashtags correctly.

In this section I gave an overview of the state-of-the-art of Twitter hashtag learning systems by mentioning different approaches that have been taken so far. Some of these systems seem to perform better than others. In later section, I demonstrate how I apply the concepts from these existing methods to develop my own hashtag classification approach.

3 Proposed Approach

I firstly try to build an algorithm to exact the features of the tweets, which is a typical NLP approach using different text cleaning method. Then, I utilize the tweets with top trends(most popular hashtags) to develop a model with supervised learning, to classify the related top trends. Finally, to evaluate the capability of the approach, I use the true top trends tweets to verify the learning performance. One difficulty involved in this procefure is dimentionality reduction due to the fact that each word is an feature: whether it's present in the document or not (0/1), or how many times it appears (an integer ¿= 0), then I applied various techniques towards this problem.

3.1 Data Collection

The dataset was constructed by requesting public tweets to the Twitter API. I have collected more than 25,000 messages during one week and, considering the worldwide usage of Twitter, tweets were only considered if the user language was defined as English. Considering that users are able to define their own hashtags, a large number of different hashtags is present in the requested tweets, there is also tremendous spam tweets sending every minutes. In order to narrow the number of classes, I decided to only consider the top trends hashtags[2]. All the messages that did not have at least one trends were discarded. In addition, Twitter API update their trends every five minutes, then my data collection also need to collecting corresponding tweets dynamically. Finally, there is about 15,000 distinct tweets with 47 distinct trends were collected, and this trend hashtag is marked as the label of each tweet for the future classification task.

3.2 Data Cleaning and Preprocessing

A tweet is represented as one of the most successful and commonly used document representation, which is the vector space model, also known as **Bag of Words**[3]. The collection of features is built as the dictionary of unique terms present in the documents collections. Each document of the document collection is indexed with the bag of the terms occurring in it, i.e., a vector with one element for each term occurring in the whole collection.

High dimensional space can cause computational problems in text classification problems where a vector with one element for each occurring term in the whole connection is used to represent a document. Also,

overfitting can easily occur which can prevent the classifier to generalize and thus the prediction ability becomes poor. In order to reduce feature space pre-processing methods are often applied. These techniques aim at reducing the size of the document representation and prevent the mislead classification as some words, such as articles, prepositions and conjunctions, called **stopwords**, are non-informative words, and occur more frequently than informative ones. These words could also mislead correlations between documents so stopwords removal technique was applied.

The **vectorizer** will then take off words that in 50% of the document, besides stopwords. The idea behind this is that if the words are so common, it may not be the feature that distinct each of the document. In other word, not have much information. I also combine this with **SelectKBest** after the vectorizer. Stemming method was also applied. This method consists in removing case and inflection information of a word, reducing it to the word stem. **Stemming** does not alter significantly the information included, but it does avoid feature expansion.

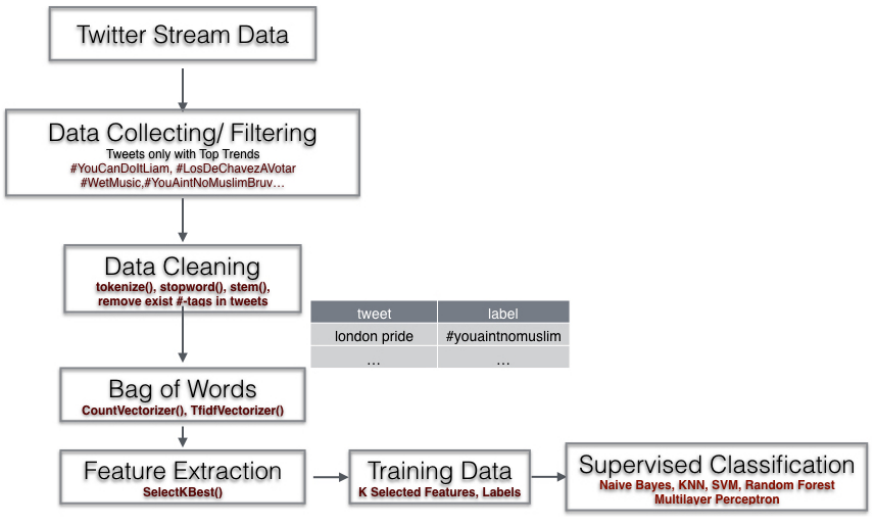

Figure 1: Proposed Approach

3.3 Supervied Classification

After being processed with bag-of-word model, the features of a tweet can be represented as a vector x where $x_i = 0$ or 1 to indicate the presence or absence of the ith featured word. Then I train and test four different supervised classifiers with 10 fold cross-validation.

- **Naive Bayes**. I firstly used a naive Bayes model to determine the relevance of hashtags to an individual tweet. Using Bayes' rule, it is can be determined that the posterior probability of C_i given the features of the vector(words presence) $x_i...x_n$ is:

$$p(C_i|x_1, ..., x_n) = \frac{p(C_i)P(x_1|C_i), ..., p(x_n|C_i)}{p(x_1, ..., x_n)} \tag{2}$$

  $p(C_i|x_1, ..., x_n)$ is the probability of using hashtag C_i given the vector of words presence, $p(C_i)$ is the ratio of the number of times hashtag C_i is used to the total number of tweets with hashtags, $p(x_1|C_i), ..., p(x_n|C_i)$ is calculated from the existing data of tweets. of tweets with hashtags

- **k-Nearest Neighbors-kNN**. In order for kNN to decide whether a document d belongs to a category c, kNN checks whether the k training documents most similar to d belong to c. If the answer is positive for a sufficiently large proportion of them, a positive decision is made.

- **Support Vector Machines-SVM**. Considering the high dimensionality of our feature vectors, which means the data is more likely to be linear separable, I was considering of conducting a linear classification. As representing the feature vectors as a sparse vector, I used scikit-learn's implementation of SVM classifier[1], which supports sparse data representations. Then I use a one-against-all strategy to make SVM a multi-class classifier.

- **Random Forest**. A random forest is an ensemble of randomly generated decision trees, each one of which will output a prediction value, in this case is the hashtag, and the final output will be the mean of all the outputs of these decision trees. Each decision tree is constructed by using a random subset of the training data by the technique of bootstrap. The Random Forest algorithm is included in scikit-learn and the parameter to tune is the number of estimators(trees).

- **Multilayer Perceptron** Finally I implemented my own Multilayer Perceptron classifier(MLP), which is an implementation of artificial neural networks. They consist of a number of nodes, grouped in different fully interconnected layers. Each node is mapping its input values into a set of output values, using an internal function. This way each variable will be processed by one or more of node trees. The outputs of the first layer will become the inputs of the second and so on until at least one node of each layer has been activated. That way each one of the variables can contribute into the final classification decision with an intelligently calculated weight. In this method, the weights between layers of a neural network are learnt using back propagation.

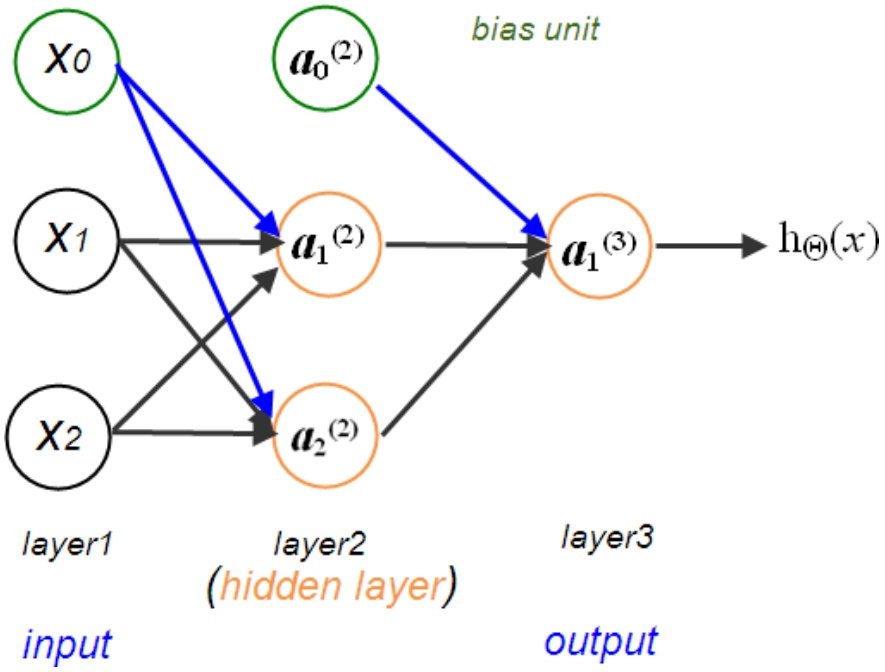

Figure 2: Multilayer Perceptron

[1] http://scikit-learn.org/stable/modules/svm.html

4 Evaluation

4.1 Test Methodology

- Data Sampling

Due to that the number of tweets with different hashtags are not uniform from the streaming data is ranging from 300 to 5000, and this distribution would significantly alter the learning performance with certain classifier. Then I firstly chose to focusing on the first **8** largest groups of tweets, the size of which is above 1000. Then relocating the order all these data to a random distribution.

- Tuning text preprocessing

When doing preprocessing on the text, I felt compelled to remove stopwords and stem on the original text. Stopwords are commonly occurring words, like "this", "that", "and", "so", "on". Is it a good decision? I don't know, so need to check. Then I first observe the performance changes with or without preprocessing(stopwords removing and stemming).

- Tuning # of Features K

Generally, there are three main approaches to Feature Selection are - Mutual Information based, Chi-square based and Frequency based[2]. There are two big univariate feature selection tools in Scikit-Learn: **SelectPercentile** and **SelectKBest**. The difference is pretty apparent by the names: SelectPercentile selects the X% of features that are most powerful (where X is a parameter) and SelectKBest selects the K features that are most powerful (where K is a parameter). Scikit-Learn provides several methods to select features based on **Chi-Squared** and **ANOVA F-values** for classification. I computed the accuracies with various feature sizes for different classifiers, using SelectKBest with ANOVA F measures.

- Validation

Validation is a cornerstone of machine learning. That's because we're after generalization to the unknown test examples. Usually the only sensible way to assess how a model generalizes is by using validation: either a single training/validation split if you have enough examples, or cross-validation, which is more computationally expensive but a necessity if there are few training points. My first step is to split the training set. Since I have 15k collecting examples, then I toke 5k for testing and left 10k for training. To avoid over-fitting, all experiment were performed using 10-fold cross-validation.

4.2 Meaturement Metrics

In this section, I will present the results of my experiments. I will start with a table presenting precision, recall and accuracy score against four different classifier(Table 1): Naive Bayes, KNN, SVM and Random Forest, with a confusion matrix(Figure 2), showing the most frequently confused hashtags of the classifier which conducted best performing. In the end, I will show the trends of performance changes by increasing the number of features selection.

4.3 Best Setting of Learning Parameters of MLP, SVM and RF

Due to the time limitation, I did not implement the measurement metrics of MLP. Furthermore, the testing on MLP is extremely complex and time consuming due to the various parameters involved. For example, all those parameters, hidden neurons in the hidden layer, number of layers, learning rate, and number of epochs, would affect the test result significantly. To minimized testing work load, I just compared MLP with other two remarkable classifiers, Random Forest and SVM, and used grid search algorithm to find the structures and parameters of learning performance(Table 2) with 1000 features selected(K).

The following parameters of MLP were examined: hidden neurons in the hidden layer={100 to 500}, number of layers={1,2,3}, learning rate={0.05,0.1,0,2,0.3}, and number of epochs={10000 to 50000}.

[2]http://nlp.stanford.edu/IR-book/html/htmledition/feature-selection-1.html

Table 1: Performance of Classifiers

Classifier	Precision	Recall	Accurarcy
Naive Bayes	0.83	0.64	0.636
Naive Bayes w/o stopwords	0.83	0.66	0.660
KNN	0.71	0.69	0.685
KNN w/o stopwords	0.73	0.69	0.694
SVM	0.82	0.81	0.813
SVM w/o stopwords	0.85	0.83	0.815
Random Forest	0.82	0.81	0.805
Random Forest w/o stopwords	0.85	0.83	0.818

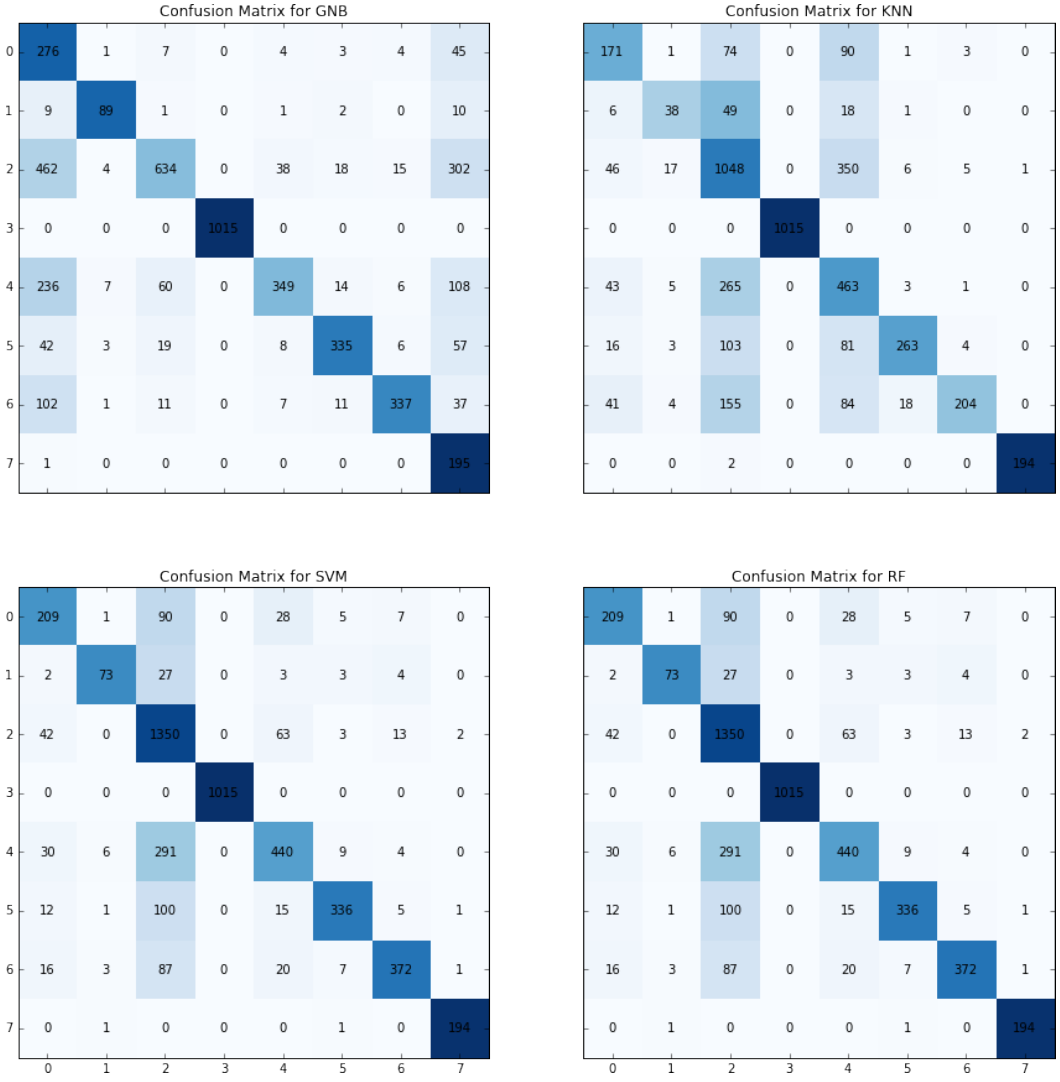

Figure 3: Confusion Matrix Performing of Different Classifiers

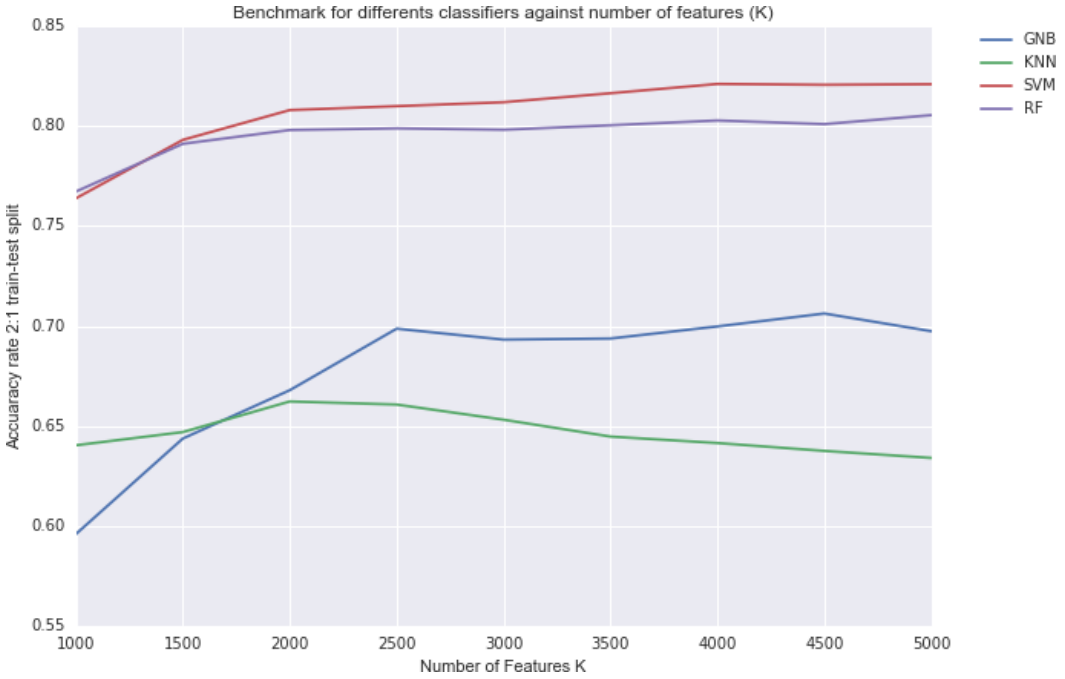

Figure 4: Test Performance by Increasing Feature Selection K

Table 2: Best Setting of Learning Parameters

Method	Parameters of the Best Performance	Acc. [%]
MLP	# of neurons= 300, #of layers= 1, learning rate= 0.1, # of epochs= 40000	82.8
SVM	Kernal Function= linear	80.2
Random Forest	# of trees= # of Selected Features(K)	79.4

4.4 Discussion

From the result shown above, the table and graph showed that the performance generally improved by removing stopwords and stemming, this can be explained that stopwords are not necessary to be considered as features for tweets classification because the text information is quite limited for each tweet with maximum 140 characters. Regarding the performance of the classifiers, SVM(with linear kernal) and Random Forest had outstanding test results(above 81%), it can be conclude that this hashtag classification for tweets is purely a linear separable task. On the other hand, the performance line chart by increasing number of features shows when K reaches above a relative large number, the accuracy score gets converge around 0.82 for the SVM classifier, which can be considered as the most robust method. Finally, the MLP method is overall the most effort costing method with a prolonged hyperparameter optimization and the performance could be altered dynamically with even a small change, but it may dominate the best performance of other classifiers occasionally.

5 Future Work

As results from our static recommendation tool have been promising, I plan to investigate the applicability of the most powerful algorithm in a real-world setting. This will involve analyzing the efficiency of this algorithm and the feasibility of real-time updates. I also plan to test the algorithm to determine how long

the learned distributions remain valid. It would be to present trending topic suggestions separately from normal suggestions to avoid overwhelming regular hashtag suggestions. Either of these strategies would require more frequent updating of the model, but this could be done as a global adjustment and need not be recomputed for individual tweets.

6 Conclusion

In this report I presented an analysis and overview of the most prominent methods for text classification of top trends tweets. The emphasis was put on the various NLP models and the combinations of various classifiers to build a robust recommendation system.

The results demonstrated the superiority of the linear classifiers(SVM, Random Forest) in micro-blogging site specifically in tweets. They also demonstrated the improvements that various combinations of NLP methods and machine learning algorithms can induce in the confidence rates of some text classification techniques.

References

[1] 150+ Amazing Twitter Statistics.

[2] To Trend or Not to Trend...

[3] Bag-of-words model, sep 2015. Page Version ID: 680036687.

[4] Fréderic Godin, Viktor Slavkovikj, Wesley De Neve, Benjamin Schrauwen, and Rik Van de Walle. Using Topic Models for Twitter Hashtag Recommendation. In *Proceedings of the 22Nd International Conference on World Wide Web*, WWW '13 Companion, pages 593–596, Republic and Canton of Geneva, Switzerland, 2013. International World Wide Web Conferences Steering Committee.

[5] Tianxi Li, Yu Wu, and Yu Zhang. Twitter hash tag prediction algorithm. In *ICOMP'11-The 2011 International Conference on Internet Computing*, 2011.

[6] Jan Pöschko. Exploring Twitter Hashtags. *arXiv:1111.6553 [cs]*, nov 2011. arXiv: 1111.6553.

[7] Kevin Dela Rosa, Rushin Shah, Bo Lin, Anatole Gershman, and Robert Frederking. Topical Clustering of Tweets.

[8] Eva Zangerle, Wolfgang Gassler, and Günther Specht. Recommending #-tags in Twitter. In *Proceedings of the Workshop on Semantic Adaptive Social Web 2011 in connection with the 19th International Conference on User Modeling, Adaptation and Personalization, UMAP 2011*, pages 67–78, Gerona, Spain, 2011. CEUR-WS.org, ISSN 1613-0073, Vol. 730, available online at http://ceur-ws.org/Vol-730/paper7.pdf, urn:nbn:de:0074-581-7.