Twitter Data Analysis

Christian Braz
George Washington University
Department of Data Science

Abstract

Today's Internet content is largely made from unstructured data, mostly images and text. Part of the text content is in form of comments made by users giving their impressions about virtually everything. Processing natural language text documents is still a challenging task due to its ambiguity and context dependency. Nevertheless, being able to extract useful information from this source is highly desirable due to its market value. Understand better people's feeling about a matter, customer needs, or product's quality, are just some of the possibilities by accomplishing this task. In this work, we evaluate the performance of machine learning and natural language processing techniques in the task of classifying sentiment in tweets.

Keywords: sentiment analysis, machine learning, natural language processing

Introduction

Since its rising in early 90s, the World Wide Web has been modifying the way people interact. Its distributed infrastructure, built upon Internet's top layers, made it pervasive and an ideal tool to enable all kinds of communications services. On a business perspective, new jargons were created trying to categorize such virtual interactions, such as Business-to-Consumer, Business-to-Bussiness, and Customer-to-Customer (Brzozowska, 2018). Social networks are also an example of interaction. Facebook, Twitter or Instagram are well known interactive platforms which enable users express their opinions.

Being able to process and extract useful information from this rich source became valuable in many different aspects and Kanhere (2011) provides a comprehensive overview. Microblogging platforms become highly successful as opinion promoter and Twitter is its main representative. Users can publish their opinions on a variety of topics, discuss current issues, complain, and express positive or negative sentiment. Therefore, Twitter is a rich source of data for opinion mining and sentiment analysis. The value of such analysis of trends is not just for marketing. For instance, Kavanaugh et al. (2012) explore the use of traditional social media content, such as Twitter, Facebook, Flickr, and YouTube, to detect, in real time, spikes in activity related to issues concerning public safety. Analyzing information from multiple social media sources should be possible to identify convergence

situations. This can be useful, for instance, to treat crisis situations, such as traffic issues, or more critical ones, like earthquakes. They developed tools like the tag cloud, which helps to identify the most frequent term in a large collection. Tag clouds, as the one in Figure 1, help the visualization of the "big picture" of social media activity.

Figure 1. An example of a tag cloud.

The purpose of this work is to evaluate two machine learning algorithms, Support Vector Machines (SVM) and Maximum Entropy (MaxEnt), for predicting two classes of sentiment, positive or negative, in tweets using embeddings as features. We try different settings for creating our own custom vectors and also using pre-trained ones. We report the overall test accuracy and f1-score of a 5-fold cross-validation training step, and also these metrics evaluated in a test set. The rest of this paper is organized as follows: Related Work introduces a brief review of the main techniques that have been used so far to address the problem of opinion classification in general and in Twitter corpora, Methods present the underlying theory behind our approach to tackle the problem. Results describe the experiment and discuss the values obtained. Finally, Conclusion presents our final considerations.

Related Work

Sentiment analysis methods

As one would expect, manual analysis of all of such prolixity through social networks is difficult and time-consuming. Sentiment Analysis (SA) (or Opinion Mining (OM)) have been introduced as an effective way to automatically extract knowledge from comments (Hemmatian & Sohrabi, 2017, Medhat, Hassan, & Korashy, 2014). More specifically, sentiment analysis is the process of extraction sentiments expressed by users in unstructured subjective texts and distinguish their polarities, whether it is a positive feeling or a negative one (Hemmatian & Sohrabi, 2017).

Medhat et al. (2014), Liu and Zhang (2012), Hemmatian and Sohrabi (2017), and Poirier, Bothorel, Neef, and Boullé (2008) roughly divide sentiment analysis into two groups: linguistic and machine learning approaches. An overall view of them can be seen in Figure 2.

Linguistic-based sentiment classification. In the research literature, opinion words are used to express desired and undesired states. Besides individual words, there

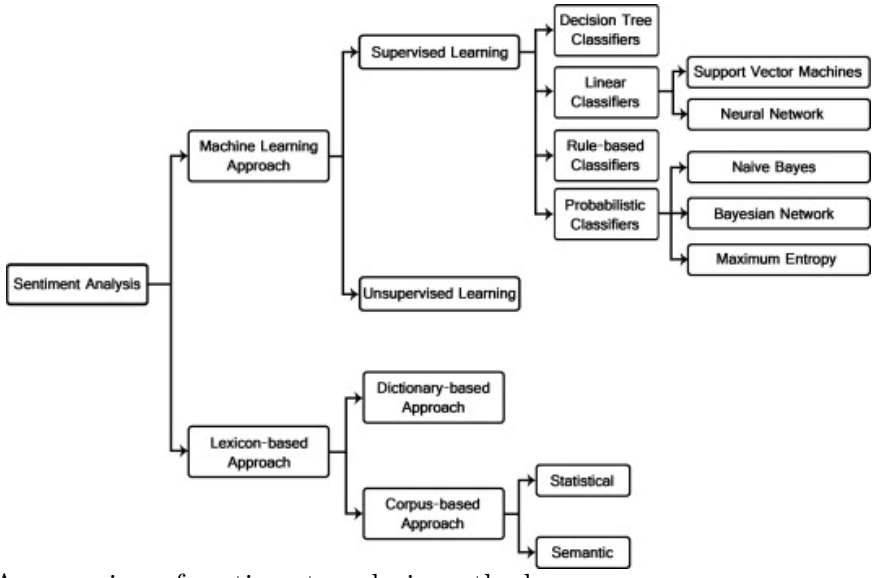

Figure 2. An overview of sentiment analysis methods.

are also opinion phrases and idioms. "Collectively, they are called *opinion lexicon* and are instrumental for opinion mining" (Liu & Zhang, 2012,p.9). Each word (or lexicon) has a polarity associated with what is just the feeling assessment that the word brings to mind (Hemmatian & Sohrabi, 2017). By studying the occurrence frequency of the word in annotated corpus of texts, word's polarity can be identified. If a word occurs more times among positive texts, its polarity is said to be positive, if it has the same frequency, it is said to be a neutral word, and if it is associated with more negative texts, it is declared to be a negative word.

There are two main strategies to find opinion words:

• Dictionary-based: this approach starts by finding a small set of opinion words with known orientation (positive, negative, neutral), the seed words. Then, it grows this set by adding their synonyms and antonyms. The iterative process stops when no more new words are found.

• Corpus-based: it tries to solve the problem that the semantic orientation of a word is domain-dependent and relies on the compilation of a set of polar words - the most suitable for obtaining a domain-dependent opinion. Its methods depend on syntactic patterns, i.e., patterns that occur together along with a seed list of opinion words to find other opinion words in a large corpus. The technique begins with a seed list of opinion words adjectives and uses them and a set of linguistic constraints on connectives (AND, OR, BUT, ...) to identify additional adjective opinion words and their orientations (Medhat et al., 2014).
Linguistic-based methods perform well in a variety of different scenarios. Its main drawback is the necessity of an extensive pre-processing step which could be expensive and time-consuming. To overcome these limitations, one can use learning-based methods which can by employed in a much more straightforward way.

Learning-based sentiment classification. Learning-based approaches mostly rely on conventional machine learning algorithms to solve text classification problems (Medhat et al., 2014). Machine learning techniques can be classified as supervised and unsuper-

vised method, and, in both cases, the aim is to build a statistical model from the data that can explain its behavior. According to Liu and Zhang (2012), "any existing supervised learning methods can be applied to sentiment classification" (p.9).

As usual for this class of algorithms, one of the first and foremost tasks is to engineer an effective set of features. The TF-IDF[1] scheme is widely used for this purpose and, as highlighted by Liu and Zhang (2012), "These features have been shown quite effective in sentiment classification." (p.10). A term can be an individual word, in which it receives the name of unigram, or longer combinations of words, leading to the n-grams schemes. Features extracted in this way are known as bag-of-words (BOW) representation because it does not rely on the order of the words, but just in their frequency. Despite its simplicity, high accuracy has been reported using unigram and bigram as features for sentiment classification.

The main drawbacks of the previous feature engineering approach is the lack in taking into account context and the size of the feature space. Recent strategies to build features from text are based on unsupervised ways to learn representations. Using the intuitive and simple hypothesis that words that occur in similar contexts tend to have similar meaning, led to the development of the new vector semantics technique Jurafsky and Martin (2000). These vectors of words embed meaningful semantic and can be used in any natural language processing application that makes use of meaning. In this work we explore the use of word vectors as features to the sentiment classification task.

Sentiment classification of Twitter

Twitter is a social networking and microblogging service that allows users to post real time messages, called tweets. Tweets are short messages, initially restricted to 140 characters and lately increased to 280. Due to the length of the messages, people use acronyms, emoticons, and other characters that express special meanings. Following is a brief terminology associated with tweets Agarwal, Xie, Vovsha, Rambow, and Passonneau (2011).

- Emoticons: These are facial expressions pictorially represented using punctuation and letters; they express the user's mood.
- Target: Users of Twitter use the "@" symbol to refer to other users on the microblog. Referring to other users in this manner automatically alerts them.
- Hashtags: Users usually use hashtags to mark topics. This is primarily done to increase the visibility of their tweets.

Pang, Lee, and Vaithyanathan (2002)provided a pioneering paper of how to apply machine learning methods such as Naive Bayes, Support Vector Machine and Maximum Entropy on the text classification problem. They used bag-of-words with unigram, bigram and part of speech as features. They concluded that using unigrams as features perform well in both classifiers and that SVM had the best performance in all scenarios. One of the early results on sentiment analysis of Twitter data is by Go, Bhayani, and Liu (2009). They used an approach called distant learning and rely on the polarity of the last emoticon in the tweet to label them as positive or negative. They built models using Naive Bayes, Maximum Entropy and Support Vector Machines, and report that SVM with unigram

[1] Term Frequency-Inverted Document Frequency

features outperforms other classifiers with 71.35% accuracy. Agarwal et al. (2011) do an extensive work designing features, based on a dictionary of emoticons and acronyms, and then they compare three models: unigram, 100 senti-features, and tree kernel model. They got a slightly better performance in the kernel (74%) and 75.4% mixing unigram + senti-features. It is worth noting that the senti-features performed almost as well as the unigram baseline (71.27%), which has about 13,000 features. A more recent result is Arslan, Küçük, and Birturk (2018) where the authors explore the use of a pre-trained word embedding model over 400 million tweets to extract the features of automatic labeled Twitter corpora and train a neural network for classification. They report much higher accuracy (roughly 85%) in a confused train/test methodology where they inadvertently use the same dataset twice.

The Experiment

The objective of this work is to compare the performance of two machine learning algorithms - Maximum Entropy and Support Vector Machine - in predicting the overall sentiment implicit in tweets using word vectors as features. The baseline is a Maximum Entropy model trained over a unigram and bigram set of bag-of-words features. The following sections depict some of these concepts and detail the whole experiment.

Dataset

We carefully selected a set of publicly available datasets to use in the experiment. Our main concern in this regards was in to find reliable datasets for training the classification algorithms. As we have not found one single large high quality file, our final dataset is a concatenation of five human annotated datasets. In resume, our datasets scheme is:

- A set of 1.6 million tweets to be used in the creation of our custom embedding. It is known as Stanford Twitter Sentiment or Sentiment140 and was created by Alec Go, Richa Bhayani, and Lei Huang, who were Computer Science graduate students at Stanford University Go et al. (2009). The sentiment of each tweet is automatically labeled (positive or negative) as opposed as having humans manual annotate. We considered that for the task of creating the embedding, in which the importance is having a large corpus that can represent the idiosyncrasies of some domain, it would be useful, but we do not rely on it to train and test the classifiers as automatic sentiment annotation of tweets using emoticons has an arguable accuracy.
- For training we select only hand annotated datasets from five different sources, four of them listed in Saif, Fernández, He, and Alani (2013), and one from Tromp and Pechenizkiy (2011).

Table 1 displays the distribution of tweets in the five selected datasets according to their sentiment labels. We can note that in the end we get a balanced dataset of 40000 examples of binary human labeled tweets sentiments.

Featute Extraction

In this section we show an overview of the feature representation techniques we use for the different models.

Table 1
Overview of the manually labeled datasets

Dataset	#tweets	#positives	#neutral	#negative
HCR	2516	541	470	1381
Sanders	5513	570	2503	654
SemEval	34183	15543	6440	12290
OMD	3238	710		1196
Tromp	11778	3458	4706	3614
	57228	20732	14119	19135

- Bog-of-words (BOW): This is the most common approach to extract features from text. The idea is to represent each doument as a (sparse) vector where each cell counts the number of times a particular word has occured in the text. BOW does not consider the order of words and it is often used with n-gram model. Just counting the number of times a word occured usualy is not enough because these raw frequences do not correspond to the actual importance of the word. To overcome this issue, the frequencies are weighted using the TF-IDF scheme. TF-IDF helps to find more discriminative words in a corpus. It depends on the frequency of a word in a document and its frequency in all the corpora. TF-IDF is calculated as follows:

$$tf\text{-}idf(t, d, D) = tf(t, d) \times idf(t) \tag{1}$$

$$idf(t) = \log \frac{n_d}{1 + df(d, t)} \tag{2}$$

- Word Semantic Vectors: word vectors, Word2Vec, word embedding are all designations of how has been known the recent supervised learning technique to learn distributed represenation of words Mikolov, Sutskever, Chen, Corrado, and Dean (2013). The skip-gram with negative sampling model encompasses the following ideas:

  – Given an input word, training a two layer neural network to predict the probability of all other words in the vocabulary be in the vicinity of this word. The input layer is a one-hot vector with the size of the vocabulary and a one in the position of the input word and zero in every other position. The hidden layer has 300 neurons with a identity transfer function, and the output is a softmax layer also with the size of the vocabulary. Given a window size which defines the vicinity, for every sentence in the corpus where the input word occurs, a randomely chosen output word is selected from the vicinity. This will be the target label for this trail which means that the neural net will learn that the input and output words are somehow related and not all the others. A scheme of this can be seen in Figure 3 [2]. At the end of the training, the hidden layer will be the word vector because one way for the network to output similar context predictions for these two words is if the word vectors are similar. So, if two words have similar

[2]From McCormick, C. (2016, April 19). Word2Vec Tutorial - The Skip-Gram Model. Retrieved from http://www.mccormickml.com

contexts, then the network is motivated to learn similar word vectors for these two words.

– From the previous explanation is easy to infer that this neural network has a huge number of parameters because it is defined in function of the size of the vocabulary. Occurs that very frequent words, like "the", do not bring useful information about the context and just overload the learning proccess. Word2Vec implements a subsampling mechanism to address this issue, and prune those words from the training. This helps in improving the quality of the representations and decrease the computanional cost.

– The 300 hidden neurons are fully conected with the vocabulary size output layer, which means update a huge number of weights in a billions of training examples. Negative sampling addresses this by having each training sample only modifying a small percentage of the weights, rather than all of them. Besides the weights of the output word, instead of update all the others, the method randomly select just a small number of negative words to update the weights. Thus, a negative word is one for which we want the network output a 0. This reduce the cost for training drastically.

To create our custom word embedding, we use the Gensim package. We choose 200 for the size of the vector and use five words as the window, given that, in average, a tweet has 50 words.

We also test using pre-trained word vectors from the Glove[3] project. Specifically, we use their 200 size word vectors trained on a Twitter corpus encompassing 2 billions tweets, 27 billions tokens, and 1.2 millions words vocabulary size.

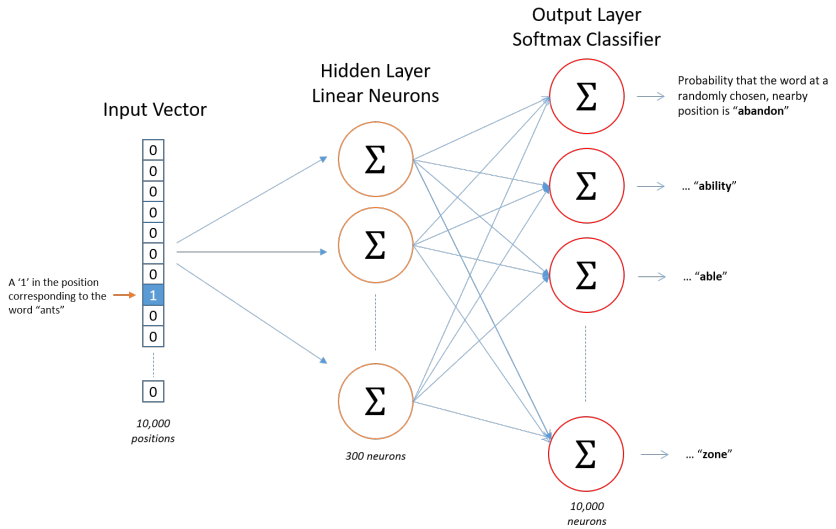

Figure 3. Skip-gram network.

[3]https://nlp.stanford.edu/projects/glove/

Classifiers

Follow a brief description of the algorithms we are using to fit a classification model.

- **Support Vector Machines (SVM):** This algorithm finds a hyperplane that separates comments according to the opinion expressed in them. The separation, or margin, is as large as possible. The algorithm returns a decision function $h(d)$, so that the probability of an opinion s given a comment d is given as:

$$p(s|d) = \frac{1}{1 + e^{ah(d)+b}}. \tag{3}$$

  where a and b are estimated by minimizing the negative log-likelihood function in $\mathcal{D}$.

- **Maximum Entropy (MaxEnt):** This is one of the most used models in a wide range of applications. MaxEnt follows this equation to find the probability of the output, given a certain input:

$$P_{ME}(c|d, \lambda) = \frac{\exp\left[\sum_i \lambda_i f_i(c, d)\right]}{\sum_{\prime c}\left[\exp\sum_i \lambda_i f_i(c, d)\right]} \tag{4}$$

Experiments

We employ a 5-fold cross-validation for training the classifiers, and report the overall accuracy and F1-score of the test. Our main experiment is designed in the following way:

• Training a logistic regression model as baseline using unigram and bigram bag-of-words features.

• Creating a custom 200 size word vectors from a 1.6 million tweets with a window of five words. Removed punctuation.

• Classification using a 5-fold cross-validation training using custom word vectors. Removed punctuation with Gensin tokenizer.

• Classification using a 5-fold cross-validation training using Glove word vectors. Preprocessing with Stanford script (kept punctuation).

Results

The results of our 5-fold test are shown in Table 2.

Table 2
Accuracy and F1-score

Algorithm	Feature	Acc	F1
Logistic Regression	BOW	0.726	0.775
Logistic Regression	Custom Word2Vec	0.762	0.820
SVM (C=1,linear)	Custom Word2Vec	0.767	0.823
Logistic Regression	Glove Word2Vec	0.783	0.834
SVM (C=1,linear)	Glove Word2Vec	0.783	0.834

The results are almost self-explanatory. We can see the advantage of using semantic word vectors in all the scenarios. We were expecting a higher accuracy when employing the

Glove embeddings since they were trained in a much bigger twitter corpus. We were also expecting a higher accuracy from SVM but both algorithms have the same performance. Figure 4 shows that our results are similar as obtained in that work.

Model	Avg. Acc (%)	Std. Dev. (%)
Unigram	71.35	1.95
Senti-features	71.27	0.65
Kernel	**73.93**	1.50
Unigram + Senti-features	**75.39**	1.29
Kernel + Senti-features	74.61	1.43

Table 5: Average and standard deviation for test accuracy for the 2-way classification task using different models: Unigram (baseline), tree kernel, Senti-features, unigram plus Senti-features, and tree kernel plus senti-features.

Figure 4. Results of paper Agarwal et al. (2011) for a binary classification.

We now show in Table 3 the results for SVM with different kernels in the custom word vectors setting. Again the regularization parameter C was kept in 1.

Table 3
SVM for different kernels.

Kernel	Accuracy
Linear	0.767
RBF	0.726
Polynomial	0.651

We can see that the linear kernel gives the best result and was the one reported in Table 2. Trying to improve the accuracy, we computed the TF-IDF value of each word and weighted with the resulting value the embedding representation of the word before averaging all of them to get the final tweet vector representation. The results are shown in Table 4.

We can note that, interestingly enough, employing TF-IDF does not lead to a better accuracy.

Table 4
Results using TF-IDF.

Algorithm	Feature	Acc	F1
Logistic Regression	Custom Word2Vec	0.743	0.812
SVM (C=1,linear)	Custom Word2Vec	0.747	0.813
Logistic Regression	Glove Word2Vec	0.757	0.821
SVM (C=1,linear)	Glove Word2Vec	0.758	0.822

Conclusion

In this work we evaluated two machine learning algorithms for classifying sentiment in tweets messages. We used word vectors, which instantiate the linguistic idea of distributional semantic, as features for the classifiers, and compare their performance with the traditional approach of bag-of-words. We used two different sets of word vectors, one that we trained and called them custom word vectors, and word vectors from the Glove project. To train the classifiers, we carefully concatenated five different manually labeled twitter corpus used in previous sentiment classification experiments.

For future experiments, we would recommend tests with higher dimensional vectors, different classifiers, and ensemble techniques for improving accuracy in a similar training approach as used here. Also, tests with neural sequence model as Recurrent Neural Networks are highly desirable as these models have been issuing the state of the art accuracy in many sequence dependent tasks.

References

Agarwal, A., Xie, B., Vovsha, I., Rambow, O., & Passonneau, R. (2011). Sentiment analysis of twitter data. In *Proceedings of the workshop on languages in social media* (pp. 30–38). Stroudsburg, PA, USA: Association for Computational Linguistics. Retrieved from http://dl.acm.org/citation.cfm?id=2021109.2021114

Arslan, Y., Küçük, D., & Birturk, A. (2018). Twitter sentiment analysis experiments using word embeddings on datasets of various scales. In M. Silberztein, F. Atigui, E. Kornyshova, E. Métais, & F. Meziane (Eds.), *Natural language processing and information systems* (pp. 40–47). Cham: Springer International Publishing.

Brzozowska, A. (2018). E-business as a new trend in the economy. *Procedia Computer Science*, *65*, 1095 - 1104.

Go, A., Bhayani, R., & Liu, L. (2009). Twitter sentiment classification using distant supervision. In *Technical report, stanford* (p. 6). Retrieved from http://help.sentiment140.com

Hemmatian, F., & Sohrabi, M. K. (2017, Dec 18). A survey on classification techniques for opinion mining and sentiment analysis. *Artificial Intelligence Review*. Retrieved from https://doi.org/10.1007/s10462-017-9599-6 doi: 10.1007/s10462-017-9599-6

Jurafsky, D., & Martin, J. H. (2000). *Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition* (1st ed.). Upper Saddle River, NJ, USA: Prentice Hall PTR.

Kanhere, S. (2011). Participatory sensing: Crowdsourcing data from mobile smartphones in urban spaces. In *Icmdm* (pp. 3–6).

Kavanaugh, A. L., Fox, E. A., Sheetz, S. D., Yang, S., Li, L. T., Shoemaker, D. J., ... Xie, L. (2012). Social media use by government: From the routine to the critical. *Government Information Quarterly*, *29*(4), 480–491.

Liu, B., & Zhang, L. (2012). A survey of opinion mining and sentiment analysis. In *Mining text data* (pp. 415–463).

Medhat, W., Hassan, A., & Korashy, H. (2014). Sentiment analysis algorithms and applications: A survey. *Ain Shams Engineering Journal*, *5*(4), 1093 - 1113. Retrieved from http://www.sciencedirect.com/science/article/pii/S2090447914000550 doi: https://doi.org/10.1016/j.asej.2014.04.011

Mikolov, T., Sutskever, I., Chen, K., Corrado, G., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th international conference on neural information processing systems - volume 2* (pp. 3111–3119). USA: Curran Associates Inc. Retrieved from http://dl.acm.org/citation.cfm?id=2999792.2999959

Pang, B., Lee, L., & Vaithyanathan, S. (2002). Thumbs up? sentiment classification using machine learning techniques. *CoRR*, *cs.CL/0205070*.

Poirier, D., Bothorel, C., Neef, E. G. D., & Boullé, M. (2008). Automating opinion analysis in film reviews : the case of statistic versus linguistic approach. In *Lrec workshop on sentiment analysis: Emotion, metaphor, ontology and terminology* (p. 94-101).

Saif, H., Fernández, M., He, Y., & Alani, H. (2013). Evaluation datasets for twitter sentiment analysis: a survey and a new dataset, the sts-gold. In *1st interantional workshop on emotion and sentiment in social and expressive media: Approaches and perspectives from ai (essem 2013)*. Retrieved from http://oro.open.ac.uk/40660/

Tromp, E., & Pechenizkiy, M. (2011). Senticorr: Multilingual sentiment analysis of personal correspondence. In *Proceedings of the 2011 ieee 11th international conference on data mining workshops* (pp. 1247–1250). Washington, DC, USA: IEEE Computer Society. Retrieved from https://doi.org/10.1109/ICDMW.2011.152 doi: 10.1109/ICDMW.2011.152