Exploration into uses of large unlabeled text datasets

Sage Hahn

12/12/17

Abstract

In this project I seek to investigate the different ways that large unlabeled datasets can be leveraged for common binary text classification tasks. In particular I seek to improve upon 'naive' strategies like keyword searches, as well as with established supervised learning tasks, my efforts constituting for the most part 'semi-supervised learning' techniques, and therefore require varying levels of user input. I present an array of techniques, both successful, but also unsuccessful, and back up each attempt with extensive experimentation. I also seek to provide insights as to why certain methods succeed over others.

1 Introduction

Only a tiny fraction of data is actually labelled, and as humans continue to amass more and more, the need for labeled data increases, as does the availability of unlabeled data. Historically it has been the case that huge datasets belonged only to large companies, governments or specialized researchers, but with the advent of large social media sites like Reddit and Twitter openly sharing data it has become possible for nearly anyone to gain access. Regardless, at least within the realm of text data, access to already labelled data is more restricted. This does not inherently restrict the ability of independent parties to do meaningful work, but it certainly makes it more difficult as most of the cutting edge machine or statistical learning techniques require large labelled datasets.

The field of 'partially supervised learning', or 'semi supervised learning' represents a promising solution when confronted with limited labelled data, and increasingly available unlabelled data. As far back as 1998 with (Nigam et al., 1998), it was show that unclassified data could be used to supplement labeled data, and to in fact increase classifier performance. Further work by (Wee Sun Lee et al., 2002) has expanded on that research, introducing novel concepts such as Expected Means with use of 'Spy documents', showing clearly that in the realm of binary text classification accurate results could be obtained with instances of only a positive training class. This article in particular overlaps significantly with the scope of my study, where the question being asked is given a text document, is it an instance of class X? Where the focus of work deviates is in making use of huge unlabelled datasets.

This notion of using unlabeled data is hardly new as mentioned above, but continues to advance such as with work by (Mann et al., 2010) exploring further the idea of semi-supervised learning. In fact this type of approach as become increasingly mainstream even appearing in Data Mining text books, e.g. **Data Mining: Practical Machine Learning Tools and Techniques 2016,** importantly though, there exists an almost endless amount of approaches. Under this assumption, I seek to approach the problem in a novel way, blending concerns around practicality, availability and ease of implementation into my research.

2 Data

Within the scope of my study I had few major sources of data. The first and largest, is a collection of reddit posts from the month of January 2015 that were made publicly available for download. These posts were read in from a flashdrive (the uncompressed file is 20gb large, and cannot fit on my computer) in createRedditTextBlocks.ipynb, where I kept only the post information (the file originally contained a great deal of supplementary data about each post. I created two “Text Blocks” of 10,000,000 posts each, and saved them as a pickled array, for ease of use in other files. The second large source of data I used was a set of Twitter, News and Blog posts originally from Coursera, though I found them via (Miyashiro, C., 2015) Each of these file contained raw text from their respective sources. Lastly, I downloaded via sklearn the ‘Twenty Newsgroup Dataset’ to use as labelled test data.

For pre-processing on all files I explored a few different techniques in testingDifferentPreProc.ipynb, but settled on applying the following changes universally in preprocessing.ipynb; breaking each document into sentences (note: the exception here is that I did not apply this step to the Twenty Newsgroup Dataset), changing everything to lowercase, and then applying the Snowball Stemmer from the python NLTK library to each word individually. Each document was then resaved in pickled form once preprocessing was complete, with their original name and Proc.pkl appended on.

I made the preprocessing steps I did for a number of reasons, but in some cases arbitrarily between two equal options. I decided to split everything into sentences because of the huge disparity in reddit comment length, with some being multiple paragraphs or lists of hundreds of usernames, to a single word. Conversely, I decided not to apply this step to the Twenty Newsgroup Data because I felt I would lose important context. With the Coursera sources data, it was somewhat irrelevant as for the most part the data had already been split into one or two sentence chunks. Further, I made the decision to Stem each word in an attempt to generalize the text from multiple sources as much as possible. I explored using lemmatization additionally, but found them to be practically quite similar, so the final choice was arbitrary.

3 Results

In order to fully explore the question of using unlabelled data, I explored two main approaches. The first, using unlabelled data from scratch, with no prior labelled data, and the second comprising the use of unlabelled data in improving accuracy of supervised approach.

3.1 From Scratch Approach

The first method I chose to explore was to see if I could improve upon one of the simplest operations, a basic keyword search. In its original inception my initial exploratory goal was to test only if such an alternative existed. The first large question to answer was therefore, what type of method is going to produce the best results? To clarify, one of the fundamental roles of the keyword search is in finding instances of a topic, call that topic X, from a set of unlabelled text documents. This then became the problem I would try to improve, one of binary text classification on a set of unlabelled documents, with baseline performance being a keyword search.

In initialExplore.ipynb I began from an unprocessed block of Reddit text (only using 1 million posts) to start to flesh out my technique. As the question I am concerned with answering involves only one class, most traditional schemes of approaching the problem do not fit quite well. I decided to answer this question, by arbitrarily selecting some ratio of data to be not class X. The idea here being to use the initial keyword search to generate cases of class X, and then to with some ratio of not class X, train a classifier to use on more data. The search I used was with keywords = ['fishing', 'caught fish', 'catching fish'], an arbitrary choice. The important new piece of information being raised at this step being the concept of the ratio of class X to not X. Without any underlying class labels at this step, I chose to simply look at how that choice in ratio related to the number of results then labelled positive by the broader classifier search. I show the results of this search in Figure 1, though notably this search doesn't being to attempt to answer the merit of the posts found.

As Figure 1 shows, the number of posts labelled positive by the classifier is highly dependant on the training ratio, and with low ratios most likely highly inaccurate. Of note is around ratio 3 or 4, where the number of posts found starts to be much closer to the baseline number from the first keyword search.

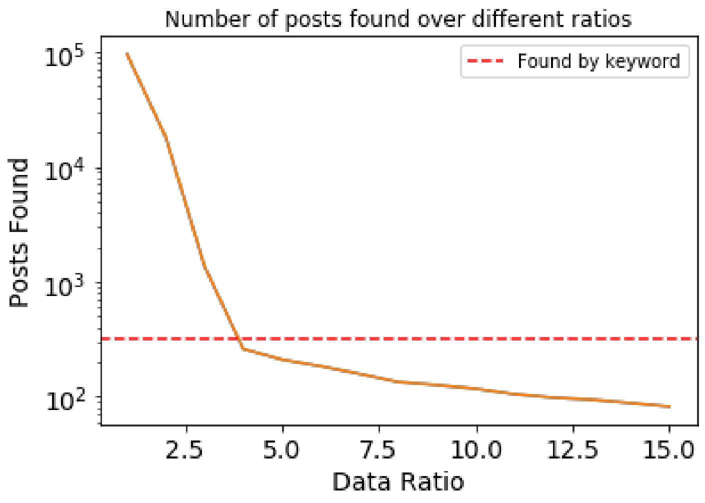

Figure 1 : The change in amount of Reddit posts classified as positive over different ratios of training data, by a simple SGD classifier. The red line representing the amount found through the initial keyword search alone.

The next big question being, how do I measure performance? In the initial exploration I ignored that question, instead focusing on rough results, but that is not nearly rigorous enough. In the next piece, explored in exploringMethods.ipynb, I chose to use the classic “Twenty Newsgroup Dataset” as my baseline labelled classes. This notably provided me fairly comprehensive text documents for 20 different classes, though notably of a different medium than the Reddit or Twitter posts. This issue of mediums overlapping also provides an interesting research question, namely do sources of text need to be of the same general type, or will they generalize? Further how does performance vary when type is similar?

To begin answering these questions, it was important to first establish a baseline accuracy, by choosing a class and keywords. I choose randomly from the classes, picking computer graphics, and modified the training and test labels to equal '1' for computer graphics, and '0' for every other class where,

computer_graphics_keys = ["graphics chip", "graphic workstation", "image coding", "digital imaging", " tiff ", "gnu plot", "computer graphics", "graphics card", "3d graphics", "thinning algorithm", "thinned image", " cgi ", "texture mapping", "curve separation", "image processing"]

As a result, I received the precision, recall and f1-score shown in Figure 2,

Keyword search for computer graphics terms

class	precision	recall	f1-score
0	0.96	0.99	0.97
1	0.60	0.18	0.28
avg / total	0.94	0.95	0.94

Figure 2 : Baseline metrics of just using a keyword search on modified Twenty Newsgroup classes

Notable in the results for the keyword search is the performance for 'class 1', or in other words, how the keyword search did on actually predicting if the sample text was about computer graphics or not. This is the case, since with significantly more cases of not class X, it would be naive to rate the classifiers performance on guessing that something is not class X. As Figure 2 shows, the precision for this method is alright at .60, but the recall and f1 score are very poor. The f1 score being calculated,

$$F_1 = 2 \cdot \frac{1}{\frac{1}{\mathrm{recall}} + \frac{1}{\mathrm{precision}}} = 2 \cdot \frac{\mathrm{precision} \cdot \mathrm{recall}}{\mathrm{precision} + \mathrm{recall}}$$

As well as often being used as a good metric of the balance between precision and recall. For the scope of analysis I will concern myself mostly with the classifiers f1 score on class 1, as I believe it most accurately reflects performance.

To attempt to beat this performance, the basic idea is to search a large unlabeled corpus of text for instances of the keywords, and to use the results at this step in the training of a classifier. The corpus used at this step is a collection of reddit posts, news, twitter and blog data, as described in the data section. Further, the corpus at this step is comprised of 32,266,806 preprocessed sentences from these varied sources. I tested a SGD classifier with the same keywords over a range of different parameters, including different loss functions, ratios of data similar to described earlier, 1-grams versus 2-3 grams, and limiting the max features selected in the Countvectorizer step. The results from this search were then sorted based on f1 score performance, as well as by precision. The top performing model with an f1 score of .37, used a training ratio of 4, the modified huber loss function, 1 to 3 grams and a max feature limit of 300 (full results in exploringMethods.ipynb at In [42])

While an f1 score of .37 isn't all that impressive, take into account that this is a statistic relative to a .28 baseline score. This experiment seems to suggest an improvement in performance is possible, but it is unclear specifically what role the composition of the unlabelled corpus held. To help answer this, I ran the experiment a few more times, once with only the News samples for the corpus, once with everything but the News samples, and lastly with a random half of the samples removed. The result from these tests are shown in Figure 3.

As can be seen in Figure 3, the relationship between different Corpus, or large unlabelled datasets, is present, but fairly minimal. The model trained with only samples from news sources preformed the best, which was expected as it is most similar, but notably it only performed slightly better. In addition I seeked to see how the size of the corpus being searched would relate to accuracy achieved, but this seemed less related then I would have expected. In conclusion, more tests would need to be done of different keywords and different samples of keywords, but the results for achieving better performance than at least a keyword are promising, though the merit of that achievement is dubious at best.

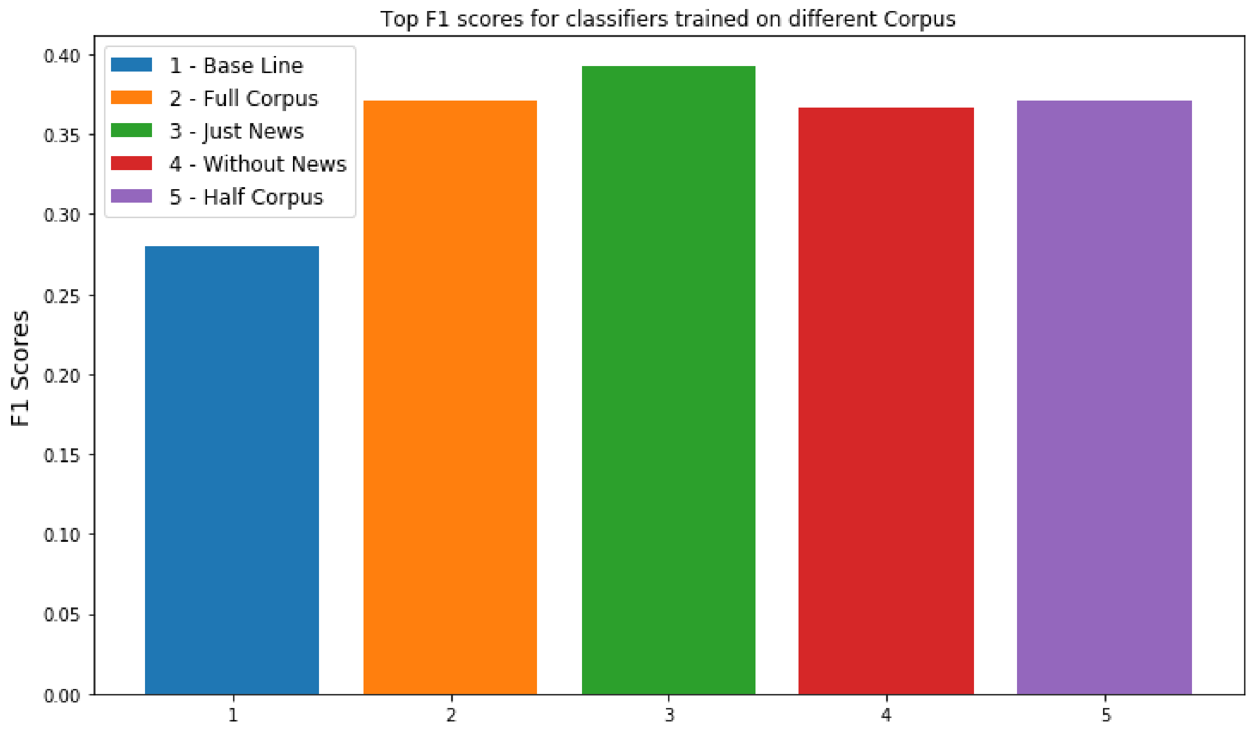

Figure 3: The top f1 scores found for classifiers trained on baseball keywords on different Corpus. Baseline f1 is also shown.

3.2 As Supplementary Data

One of the most documented uses of unlabeled datasets is in their use as supplementary data. What this means, is that given an already established supervised learning problem, in this case the Twenty Newsgroup Dataset, can I improve baseline accuracy with access to an unlabeled dataset? As in the earlier section, before I can begin to answer this question I need to establish a baseline approach. The most common use of the dataset in training a multiclass classifier to distinguish between the different subjects, but in my case the concern is shifted to focusing on only one class.

To begin, I established my initial method as simply redefining all classes except one to be 0, thereby redefining the problem as one of binary classification (in exloringMethods.ipynb). I next defined an SGD classifier, with count vectorization and a TfidfTransformer, and tested it on a small sample of the data. I determined from here that it would be worthwhile to define a test over the different ratios of test data about class X, to not class X, similar to the problem investigated in 3.1. I present the averaged results for f1, precision, and recall scores over all twenty classes in Figure 4. With the baseline performance established at a little under .8 f1 score with a ratio of 2 (note as an average across all classes) I could then test additional methods. I explored two different main techniques in this section.

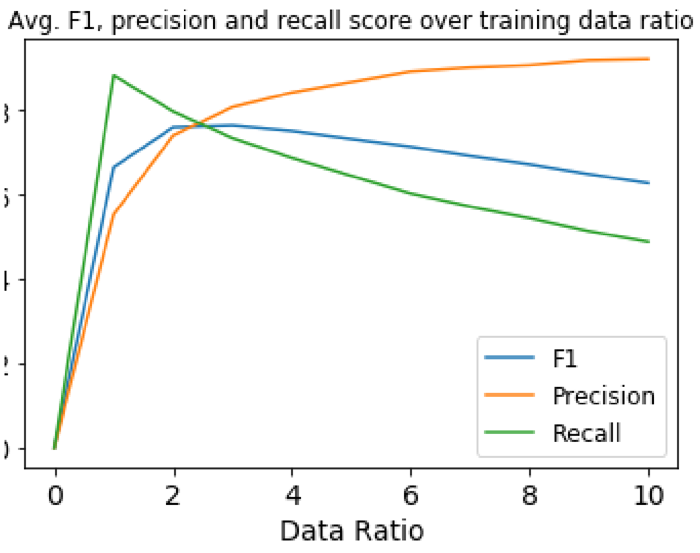

Figure 4: The average F1, Precision and Recall scores over all twenty classes, with default SGD classifier, Counter Vectorizer and Tf-Idf settings.

The first technique explored involved choosing a class, and then finding strongly associated keywords similar to in 3.1, though the difference here being I have a discrete set of training data to help me decide on good keywords. I define a helper function getKeyWords, in exploringMethods.ipynb (In [2]), which returns for a class, and specific ratio, the top ten 1-gram and 2-gram words that appear most for that class, as well as appear least in the sampled other classes. I found this method produced a good idea of the most common keywords, in addition to some garbage punctuation, and other outliers, which meant human input would be required at this step.

The next step of this process takes the keywords generated by me, or the user, and searches the corpus as before. The difference lies in instead of training a classifier from only the lines with keywords, to instead supplement the initial training data with the keywords. From here, a classifier is trained on the now augmented training data, and then is validated with the validation set. A grid search is then run over a number of different hyperparameters, including again notably, the ratio of training data. One important decision had to be made at this step, of whether to add in additional samples of class not X from the corpus into the training data, or to add samples of not X from the original training data. I settled on adding from the corpus due to logistical reasons.

For my first run through of this method I, using getKeyWords, determined the key words for the class of news articles on baseball, to be,

baseball_key_words = ['basebal', 'the cubs', 'red sox', 'major leagu', 'the yanke', 'the philli', 'the brave']

These keywords were chosen based on frequency, as well as deemed specific enough to not unintentionally generate other results. A search with these keywords over different classifiers yielded a best f1 score of only .67 (exploringMethods.ipynb In[64]) which is lower than the baseline. Before

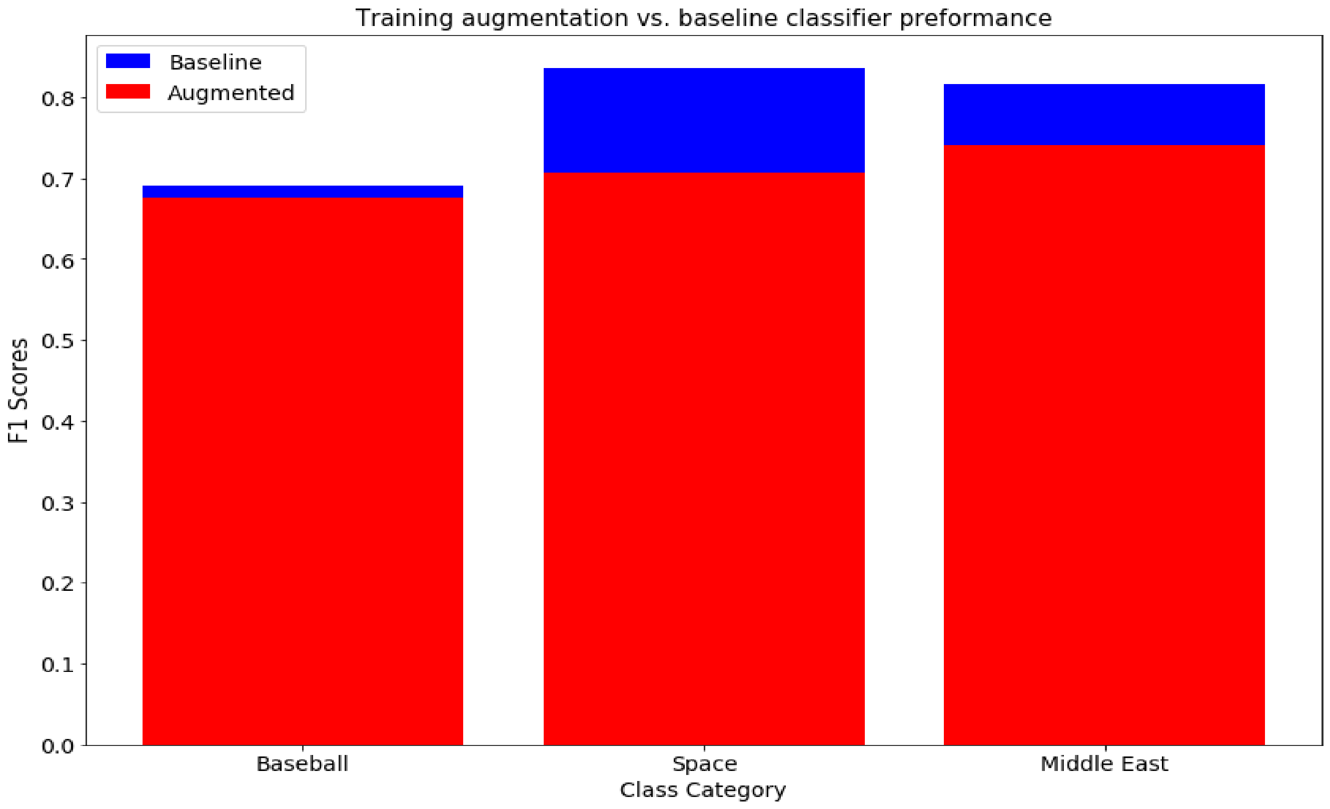

Figure 5: Comparing performance of best f1 score found for augmenting classes with keyword search results compared to baseline.

discounting this method as flawed, I tested again with only the keyword 'basebal' (Note not a typo, this is the stemmed version of baseball), and got X (In [67]), still under expected performance. Regardless, I tried it further on articles about space, as well as news on the middle east, but continued to receive frankly awful results,which can be seen in Figure 5.

My other proposed technique for increasing performance on an established supervised learning problem is quite similar my former strategy, with one key difference. With this next technique, I run a search over possible classifier using just the given training data, but this time I am looking for a classifier with very high precision. Once found, the idea is that high precision should ensure that instances found are actually part of class X, and then, if working properly, to add these instances into the training data, and to retrain from there.

I ran into roadblocks with this technique right away (see the end of explingMethods.ipynb, In [94]). I found a good model with close to a precision score of 1, and decent recall using the Middle East news class, and then ran it to predict on a chunk of a 6 million lines from the corpus (any more would give me memory errors). This approach found ~2000 lines which it predicted as part of the middle east class, which I then added to the training data, re-trained the classifier, and found fairly stagnate, but slightly decreased performance. To diagnose the problem, I looked into some of the samples the classifier was labelling a class 1, and found them to be for the most part accurate. The classifier turning out predictions that seemed to me to be anecdotally about the Middle East. In short, this idea seems to work at least anecdotally, but I did not have have enough to time to come up with a good way to test its accuracy. Just because it didn't increase the performance on the news set, doesn't mean it isn't working.

As a final step I explored using the word2vec approach as described in https://blog.keras.io/using-pre-trained-word-embeddings-in-a-keras-model.html (my code in word2VecApproach.ipynb). My goal at this step was to test the binary classification performance of the Twenty Newsgroup Dataset with a different form of auxiliary data, namely the gloVe 200-dimensional word embeddings (Pennington 2014). This approach transforms words into 200 dimensional vectors trained on a large Wikipedia dataset from 2014, and contains the vector representations for 400,000 different words. As this involves a different type of preprocessing, I used the raw data from sklearn for this section.

Using 20% of the data for validation, and a basic convolutional neural network architecture for binary classification (last layer sigmoid function, with binary cross entropy loss function). I was able to get an f1 score of .94, precision score of .92, and recall of .96. While not quite the method I set out to prove, this last effort shows that using pre-trained general data as word embeddings, can significantly increase performance.

4. Discussion

This foray into uses of unlabeled data was not quite as fruitful as I initially envisioned. I for the most part worked to enact ideas which I thought might produce interesting results, and especially in the case of augmenting a supervised learning problem seemed to fail. In the other case discussed in 4.1, while I did achieve my goal of beating the baseline performance, I don't think my method of beating it was all that better. In the future I would like to investigate how different unsupervised learning algorithms like spy - EM, or even clustering algorithms could work on the problem posed in 4.1.

My exploration of augmenting performance in 4.2 did stumbleupon another interesting field of inquiry though, that being word2vec. While it wasn't the method of augmentation I imagined, I was able to fairly quickly get very impressive results. With more time I would have liked to investigate if I could come up with a good way to adapt the question in 4.1 to work with a word2vec approach. If for example, some keywords about a subject, or even lines of example text could be used to search through a corpus, finding similar instances. While this type of approach didn't work how I originally implemented it, perhaps with the impressive generalizing power of pre trained word embeddings, I could manage some better results.

In conclusion, this research question is not easily answered, but other work in additional to mine here, seem to suggest it is possible to use unlabelled dataset. It just might very well be the case that in order to get good results with these datasets you need much much larger sizes, and computational resources, for example in the creation of gloVe. Luckily, as part of the scope of the problem I was concerned with was the ability to answer these types of questions with limited time and resources, these pre trained word embedding networks are becoming more readily available. The general trend of transfer learning, at least to me, appears to be one of the more promising areas of research and development.

5. Works Cited

-Mann, Gideon S. and McCallum, Andrew. Generalized expectation criteria for semi-supervised learning with weakly labeled data. Journal of Machine Learning Research, 11:955{984, 2010.

-Miyashiro, C. Text Mining and Natural Language Processing, 2015 http://rstudio-pubs-static.s3.amazonaws.com/67435_ca0769f0dbbb4fc4bda5e4535e21fb54.html

-Nigam, K., McCallum, A., Thrun, S., & Mitchell, T. (1998). Learning to classify text from labeled and unlabeled documents. AAAI-98 (pp. 792–799). Madison, US: AAAI Press, Menlo Park, US.

-Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation.

-Wee Sun Lee, Bing Liu, Philip S. Yu, Xiaoli Li, " Partially Supervised Classification of Text Documents", Machine Learning-International Workshop And Conference-, 2002