Novel Approach for Semi supervised learning using Label Propagation and Support Vector Machines

Abstract

Semi supervised learning methods have gained importance in today's world because of large expenses involved in labeling the unlabeled data manually. The proposed approach uses SVM and Label Propagation to label the unlabeled data. In the process, at each step SVM is trained to minimize the error and thus improve the prediction quality. The approach is tested using 12 datasets of different sizes ranging from order of 1000s to order of 10000s. Results show that the proposed approach outperforms Label Propagation alone by a large margin with F-measure of almost twice on average as compared to Label Propagation.

1. Introduction

In many real world scenarios, the proportion of labeled and unlabeled data is very skewed. This is mainly because labeling the unlabeled data is labor expensive and time consuming. It is mainly for this reason that semi supervised approaches are preferred over supervised ones despite knowing that latter will perform better than the former. Semi supervised learning model can either be built by first training the model on unlabeled data and using labeled data to induce class labels or vice versa.

Semi supervised learning methods are mainly classified into two broad classes: Transductive and Generative. The goal of transductive models is to label the given unlabeled data in the best possible way. They seek to find out the probability $P(Y|X)$ where Y is class label and X is the feature vector. Gaussian processes, transductive SVM and graph-based methods fall in this category. On the other hand, Generative models are designed to label the unlabeled records from the whole sample space. These methods are based on joint distribution and examples include Expectation Maximization.

Transductive and Generative models are often explained using following analogy: Transductive learning is compared to the situation when a student learns concepts and wants to perform well just in the given home assignment. On the other hand, when a student prepares so as perform well on ANY unseen question, it is analogous to Generative learning.

The proposed approach labels the unlabeled data using Label Propagation and SVM. At every step in the process, it fits the model to minimize error and thus improve the prediction quality.

The remainder of the paper is organized as follows: Section 2 discuss the related work and the proposed approach is presented in section 3. Section 4 contains the experimental results and comparison of the proposed approach with the Label Propagation algorithm followed by Conclusion and future work in section.

2. Related Work

A lot of work has been done in the field of semi-supervised learning. It is clear that in every efficient approach, unlabeled data has to be play the major role which is in large amount as compared to labeled data. (Castelli & Cover, 1995) showed that unlabeled data can predict better if the model assumption is

correct. But if the model assumption is wrong, unlabeled data may actually hurt accuracy. Since then a lot of new algorithms have been designed to make use of abundant unlabeled data. Nigam et al. (2000) applied the Expectation Maximization algorithm on mixture of multinomial for the task of text classification and showed the resulting classifiers predict better than classifier trained only on labeled data.

A commonly used technique for semi-supervised learning is self-training. In this a classifier is initially trained with the small quantity of labeled data. The classifier is then used to classify the unlabeled data and most confident points are added to the training set. The classifier is re-trained and the procedure repeated. Word sense disambiguation is successfully achieved by Yarrows (1995) using self-training. Subjective nouns are identified by Riloff et.al (2003).Parsing and machine translation is also done with the help of self-training methods as shown by Rosenberg et al. (2005) in detection of object systems from images.

A very effective way to combine labeled data with unlabeled data is described by Xiaojin Zhu et. al. which propagated labels from labeled data points to unlabeled ones. An approach based on a linear neighborhood model is discussed by Fei Wang et. al which can propagate the labels from the labeled points to the whole data set using these linear neighborhoods with sufficient smoothness.

3. Proposed Approach

Pseudo Code

Serial Algorithm

Input: Classifier, Threshold
Output: F-measure

1. (labeled_records, unlabeled_records) = select_next_train_folds()
   # Each fold of data is split into labeled and unlabeled records with 20 : 80 ratio
   # unlabeled_records have their class field set to -1
2. test_records = select_next_test_fold()
   # Concatenate labeled and unlabeled records to get train_records
3. train_records = labeled_records + unlabeled_records
4. newly_labeled = 0
5. **while** len(labeled_records) < **len**(train_records)**:**
    a. lp_probability_matrix = run_lp(labeled_records + unlabeled_records)
    b. model = fit_classifier(classifier, labeled_records)
    c. labeled_atleast_one = **False**
    d. for record in unlabeled_records:
        i. classifier_out = model.predict_class(record.feature_vector)
           # Test for LP and classifier agreement
        ii. **if** lp_probability_matrix[record.feature_vector][classifier_out] > threshold:
            1. unlabeled_records.remove(record)
            2. record.class_label = classifier_out *#label the record*
            3. labeled_records.add(record) *#add the newly labeled record to set* of *labeled records*

4. newly_added += 1
   # Set labeled_atleast_one flag to True if at least one new record is labeled in current iteration of while loop
5. labeled_atleast_one = **True**

 # Break the loop if no new record is labeled in current iteration of while loop

e. **if** labeled_atleast_one == **False**:
   i. **break**

 # Compute F-measure of constructed model

6. test_records_features = test_records.get_feature_vectors()
7. test_records_labels = test_records.get_labels()
8. predicted_labels = model.predict(test_records_features)
9. f-measure = compute_fmeasure(predicted_labels, test_records_labels)

Parallel Algorithm

Input: Classifier, Threshold, No_of_tasks *#Number of parallel processes*
Output: F-measure

5. newly_labeled = 0
6. **while** len(labeled_records) < **len**(train_records):
   a. lp_train_records = labeled_records + unlabeled_records
   b. lp_probability_matrix = []; classifier_out = []
   c. lp_process = new_process(target = run_lp, args = (lp_train_records, lp_probability_matrix))
   d. lp_process.start()
   e. classifier_process = new_process(target = fit_classifier, args = (classifier, labeled_records, unlabeled_records, classifier_all_out))
   f. classifier_process.start()
   g. lp_process.join()
   h. classifier_process.join()
   i. atleast_one_labeled = **False**
   j. chunk_size = **len**(unlabeled_records) / No_of_tasks
   k. all_pids = []
   l. None_initialize(labeled_lists, No_of_tasks)
   m. None_initialize(unlabeled_copies, No_of_tasks)
   n. **for** i in **range**(**len**(labeled_lists)):
      i. start = i * chunk_size
      ii. end = (i+1) * chunk_size
      iii. unlabeled_copies = unlabeled_records[start : end]
      iv. lp_probabilities = lp_probability_matrix[start : end]
      v. classifier_outs = classifier_all_outs[start : end]
      vi. label_records_process = new_process(func = label_data, args = (unlabeled_copies[i], labeled_lists[i], lp_probabilities, classifier_outs, threshold))
      vii. label_records_process.start()
      viii. all_pids.append(label_records_process)
   o. unlabeled_records = []

p. done_processes = []
q. **while len**(done_pids) < **len**(all_pids):
    i. **for** i in **range**(**len**(all_pids)):
        1. **if not** all_pids[i].is_alive() **and** (i **not in** done_pids):
            a. done_processes.append(i)
            b. unlabeled_records += unlabeled_copies[i]
            c. labeled_records += labeled_lists[i]
r. **if** atleast_one_labeled == **False**:
    i. break

 # Compute F-measure of constructed model

7. predicted_labels = []
8. test_records_features = test_records.get_feature_vectors()
9. test_records_labels = test_records.get_labels()
10. run_parallel_classifier(predicted_labels, labeled_records, test_records_features, classifier, no_of_tasks)
11. f-measure = compute_fmeasure(predicted_labels, test_records_labels)

Description: Serial

The pseudo code enumerates steps taken during one iteration of n-fold cross validation. Lines 1 and 2 select and assign folds to training and test data respectively. Training data now contains 80% unlabeled data and 20% labeled data. Line 5.a runs LP on labeled and unlabeled data and fetches probability matrix for unlabeled data. Line 5.b runs the classifier on labeled data. Lines 5.d.ii.1 to 5.d.ii.5 update unlabeled data list and labeled data list if it finds any unlabeled record satisfying following condition: LP probability for the class predicted by classifier on that record is greater than threshold. Such a record is first removed from unlabeled data list (5.d.ii.1), labels it according to output of classifier (5.d.ii.2) and adds it labeled data list (5.d.ii.3). Control variables (newly_added, labeled_atleast_one) are also updated. Line 5.e breaks the loop if no records are labeled in the current iteration of while loop. The while loop can continue till all the unlabeled records are labeled. Lines 5-9 compute F-measure of the trained model on the test set.

Description: Parallel

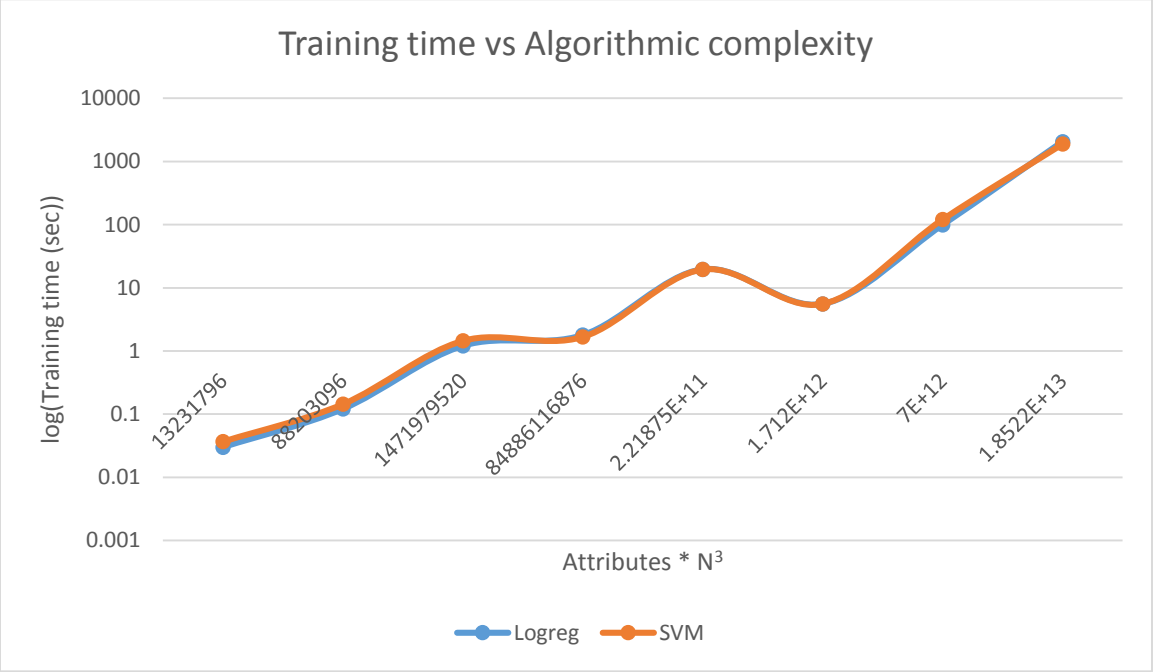
We have introduced concurrency in the algorithm at a very coarse level. Pseudo code for parallel algorithm is exactly same till step 5 as compared to serial algorithm – which involves preparing training and test sets. Lines 6.c to 6.d create and start new child process for Label Propagation. Lines 6.e and 6.f do the same for the classifier. Main unit waits for both of these processes to get over on lines 6.g and 6.h. Both of these processes fill the empty lists which are passed to them (lp_probability_matrix and classifier_all_out) with Label Propagation probabilities and predicted class labels on unlabeled data respectively. 6.j calculates the chunk_size - number of unlabeled records to be sent to each process for processing. Chunks are extracted from Unlabeled records, Label Propagation probabilities and classifier outputs on 6.n.iii to 6.n.v. Each of these chunks has length chunk_size. These 3 chunks along with threshold and an empty list (labeled_list[i]) are sent to a new child process for processing. This process shall move each record eligible for labeling from unlabeled_copies to labeled_list[i] after labeling it. Line 6.q processes the outputs of the processes. It keeps iterating over list of newly created processes and processes their outputs as soon as any of them is finished. It builds new unlabeled_records and labeled_records list by putting together all the unlabeled_copies and labeled_lists[i] respectively (both of which are modified by the child processes). The flag atleast_one_labeled is a shared among all the

processes and its access is regulated using a lock. The while loop on line 6 shall break either if all the records are labeled or no records are labeled in an iteration (value of atleast_one_labeled is False). Lines 7-11 compute F-measure. run_parallel_classifier function passes chunks of test_records to a new child process along with the model object for predicting its classlabel. These chunks are built using method very similar to that on lines 6.n.iii to 6.n.v. fit_classifier method (used on line 6.e) also similar parallel method as used by run_parallel_classifier for predicting class labels of the unlabeled records. Finally f-measure is calculated in the traditional way on line 11.

4. Experimental Section

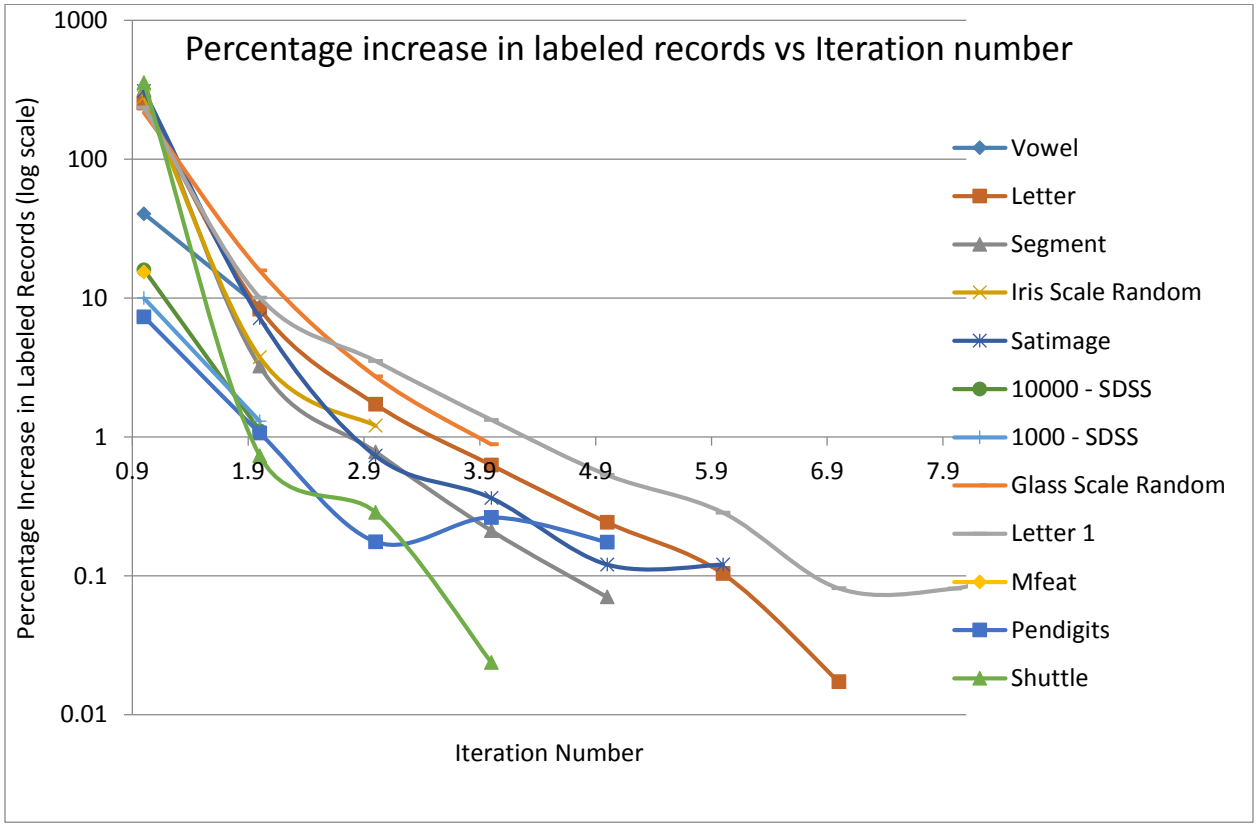
We considered 12 different datasets for our experimental runs. The name of the datasets with their number of attributes (excluding the class label) and records are as follows (in the format – *dataset : (no of attributes, no of records)*): Vowel: (10, 528), Letter: (16, 10500) , Segment: (18, 2310), Iris scale random: (4, 149), Satimage: (36, 1331), 10000 – SDSS : (7, 10000), 1000 – SDSS : (7, 1000), Glass scale random : (9, 214), Letter 1 : (16, 4500), Mfeat : (214, 2000), Pendigits : (16, 7494) and Shuttle : (9, 12770). We ran the algorithm for different values of Threshold, Classifier and percentage of initially labeled data in the training set on each dataset. For each run, we noted values of Training time, Percentage increase in labeled records for each iteration, Percentage of labeled data at the end of the run and F-measure. For comparison purposes, we also noted the F-measure of Label Propagation algorithm for each run. We also ran the parallel algorithm for each dataset with best F-measure producing values of input parameters and noted training time. To show that speedup persists for large data sizes, we ran parallel algorithm for datasets containing different number of records (but same number of attributes) and noted the decrease in training time. Following graphs show the results:

Fig 1



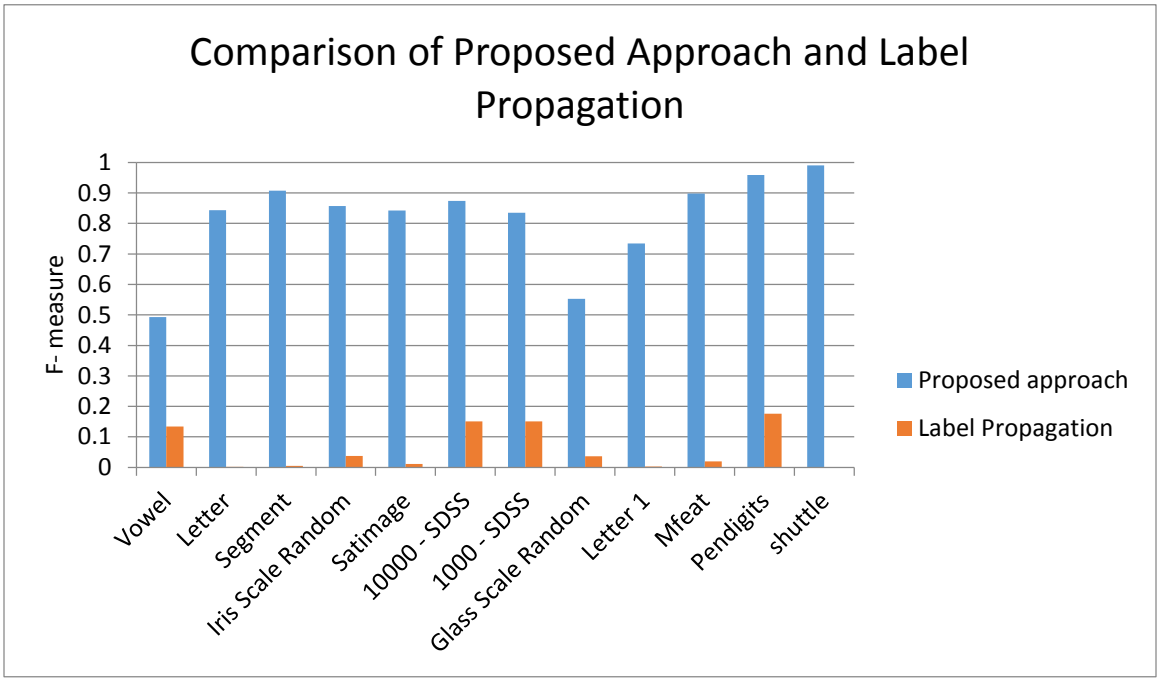
To see the effect of the dimension and the number of records in the dataset, we plotted Training time with respect to the cube of number of records in the dataset for the best performing classifier and threshold. The axis were chosen by keeping in mind the O (dim * N^3) complexities of both SVM and Logreg. The graph tends to show polynomial increase in training time as N increases (instead of linear). This may be the effect of neglecting lower order terms in the complexity expression of SVM and Logreg.

Fig 2



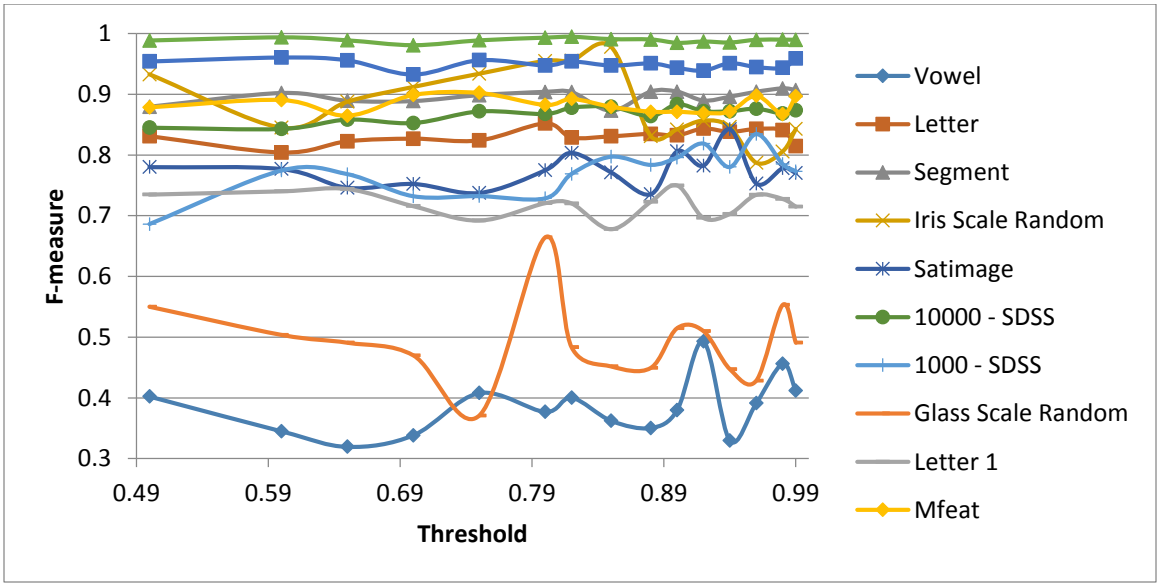
We plotted Percentage increase in labeled records as compared to previous iteration for each iteration for the best choice of classifier and threshold (according to F-measure). The graph shows that percentage increase in labeled records decreases exponential (note the log scale) as the iterations progress. This means not much data gets labeled as loop progresses. This is the consequence of Label propagation and SVM not agreeing on deciding the class label which is to be assigned to the unlabeled record. While labeling an unlabeled record, there is low chance of SVM getting wrong since it is always trained on labeled data. This means that the quality of labeling done by Label propagation decreases significantly as iterations progress. This deterioration in Label Propagation's quality has very little effect on algorithm's overall prediction quality because of the correct predictions done by SVM at every step while labeling the unlabeled records and thus we get better performance than Label propagation.

Fig 3



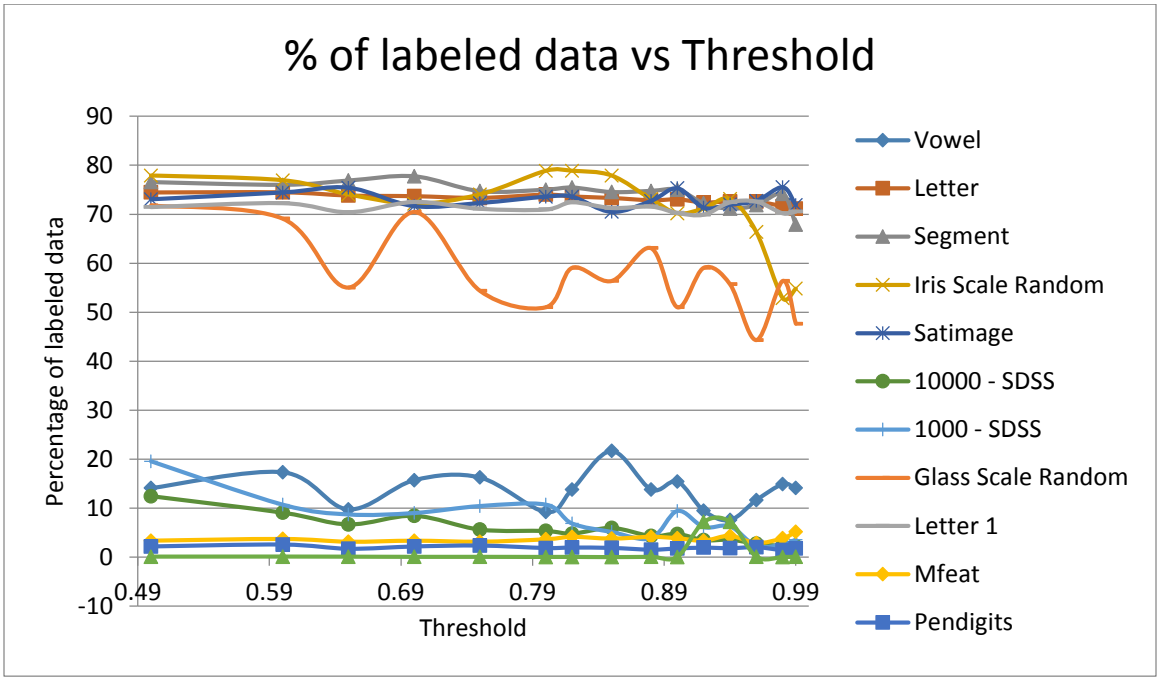
The graph compares the performance of the proposed approach and the Label propagation algorithm for all the datasets. F-measures of the proposed approach were noted for best choice of classifier and threshold. For Label Propagation, all the unlabeled records were labeled according to the class corresponding to highest Label Propagation Probability. In all the cases, the proposed approach outperforms Label Propagation by a large margin. This high quality of performance can be attributed use of SVM together with Label Propagation to label the unlabeled examples. No unlabeled example is labeled without the agreement of both – SVM and Label Propagation. This significantly reduces the pitfalls caused by Label Propagation increasing the prediction quality of overall approach.

Fig 4



Graph was plotted for different values of Thresholds considering best choice of classifier. It shows that F-measure and Threshold are poorly correlated. Intuitively, as threshold increases, data labeling process should become stricter and commit less mistakes. Thus precision should increase and recall should decrease, decreasing F-measure (Graph of Precision, Recall vs. Threshold required here?). But this is not the case. This can be reasoned as follows: Decrease of Label propagation probability threshold results in increase in number of unlabeled records being considered for labeling. But this does not increase number of records eventually labeled at similar rate. This is because a record can be labeled only when SVM and Label Propagation agree on the class label. So, unlabeled records marked by label propagation for labeling with low confidence are discarded by output of SVM. Thus percentage of labeled data at end of the run fluctuates very little for all the Thresholds (as it is seen in next graph). So change in thresholds, has little effect on F-measure of the algorithm.

Fig 5



Graph of Percentage of labeled data was plotted for different values of Thresholds considering best choice of classifier. It shows that percentage of labeled data and Threshold are poorly correlated. Intuitively, as threshold increases, data labeling process should become stricter and label less number of records. So, the graph should be downward sloping curve. But this is not the case. This can be reasoned as follows: Decrease of Label propagation probability threshold results in increase in number of unlabeled records being considered for labeling. But this does not increase number of records eventually labeled at similar rate. This is because a record can be labeled only when SVM and Label Propagation agree on the class label. So, unlabeled records marked by label propagation for labeling with low confidence are discarded by output of SVM as threshold increases. Thus percentage of labeled data at end of the run fluctuates very little for all the Thresholds (as it is seen in next graph). So change in thresholds, has little effect on F-measure of the algorithm.

Above two graphs bring about another feature of the algorithm. For the datasets: 1000 records - SDSS, 10000 records –SDSS, Meat, Pendigits and Shuttle, F-measure Percentage of labeled data is very low 0-20 %. But there F-measures are reasonably high between 0.67 to 0.9. This shows that high F-measure does not always require high amount of unlabeled data to be labeled. As long as the algorithm is able to label representative records in the dataset, it is likely to give good F-measure.

Fig 6.1

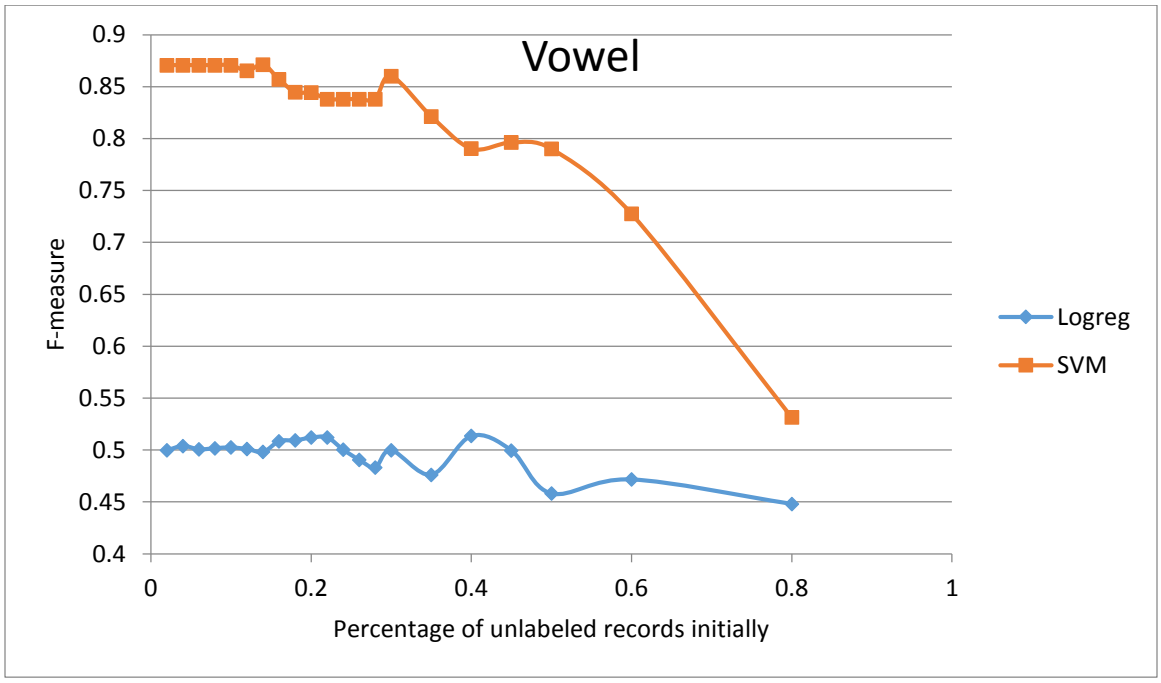


Fig 6.2

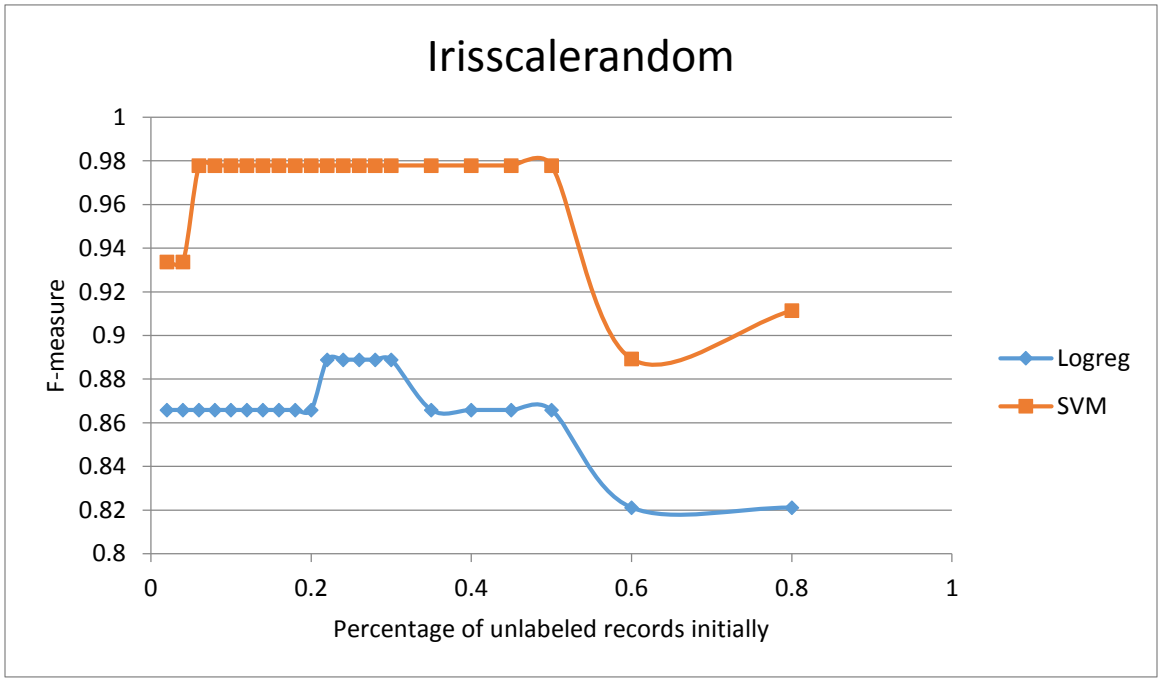


Fig 6.3

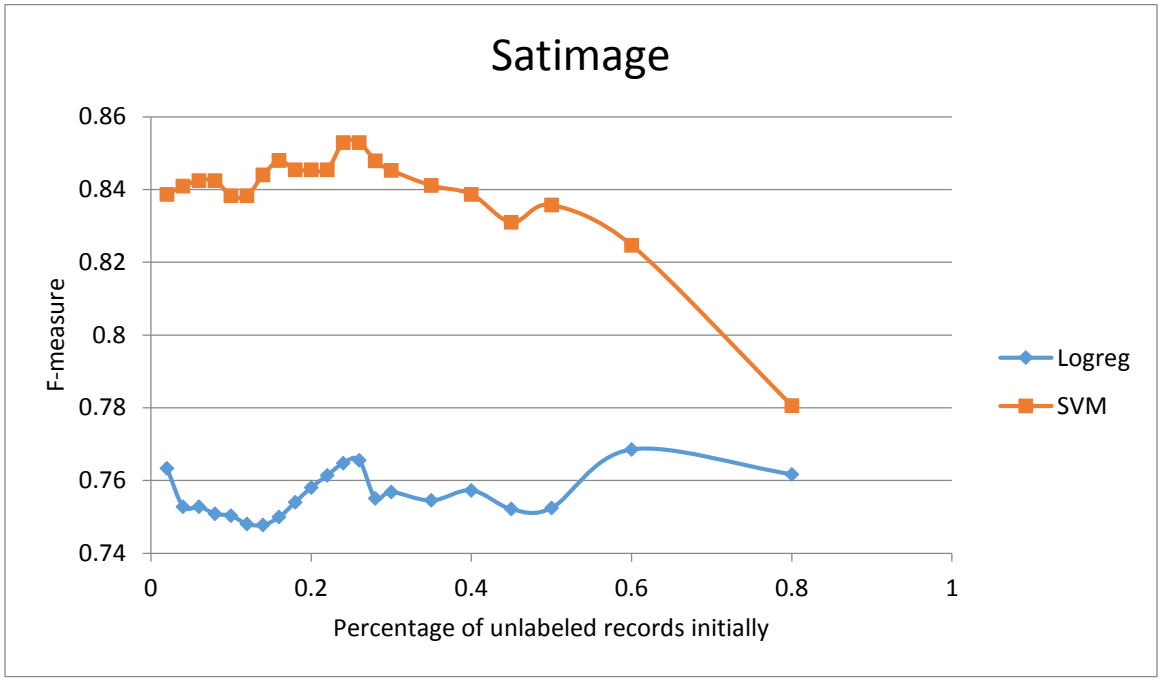

Satimage
0.86
0.84
0.82
0.8
0.78
0.76
0.74
F-measure
0
0.2
0.4
0.6
0.8
1
Percentage of unlabeled records initially
Logreg
SVM

Fig 6.4

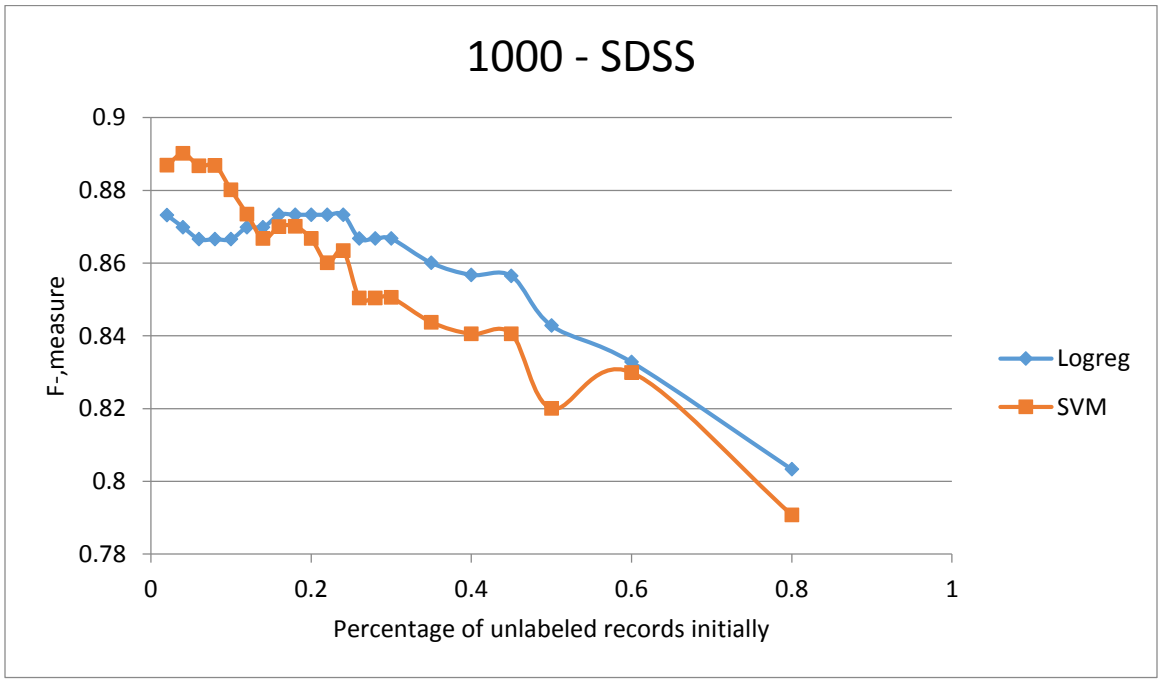

1000 - SDSS
0.9
0.88
0.86
0.84
0.82
0.8
0.78
F-,measure
0
0.2
0.4
0.6
0.8
1
Percentage of unlabeled records initially
Logreg
SVM

Fig 6.5

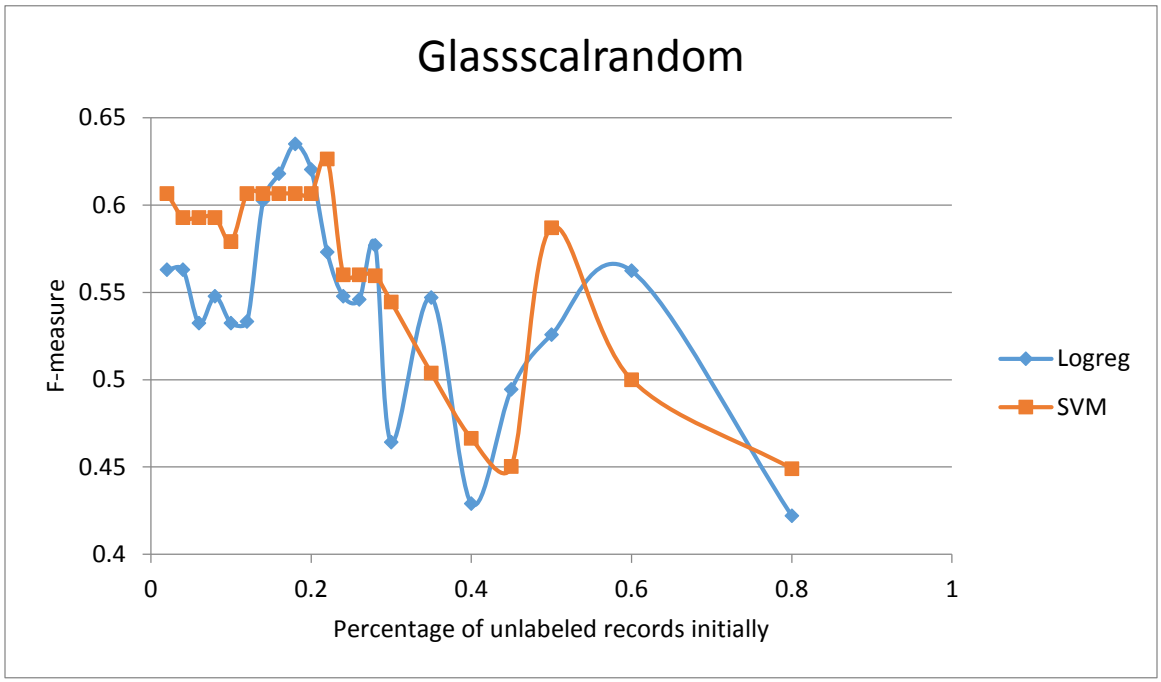
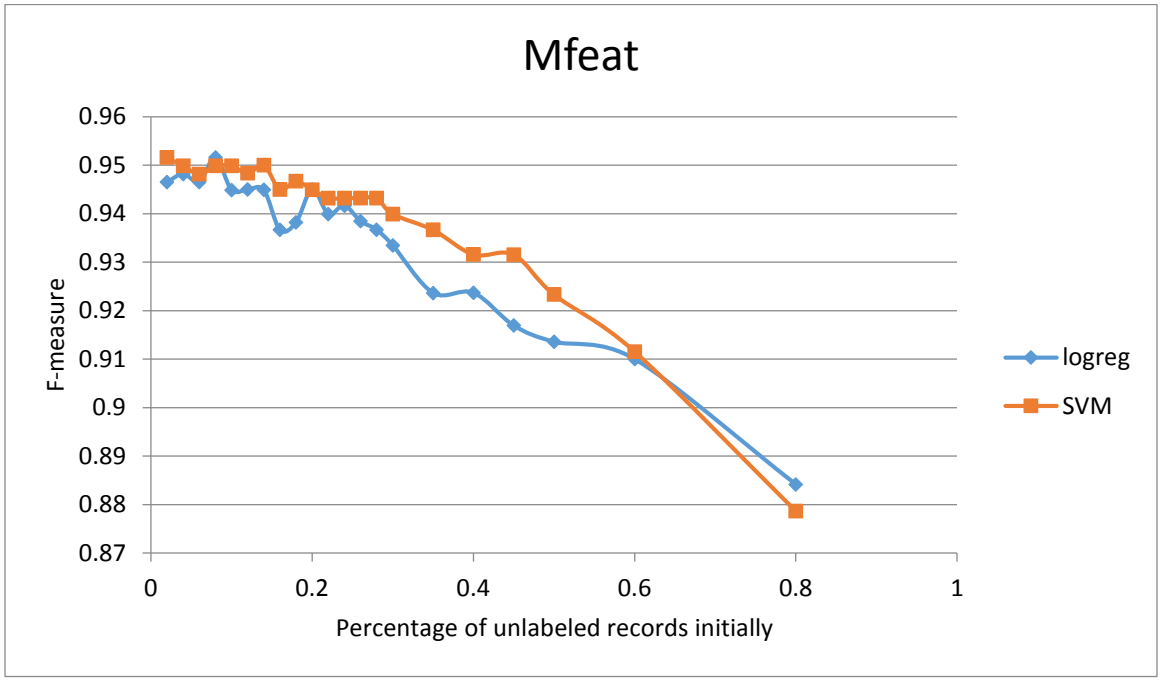


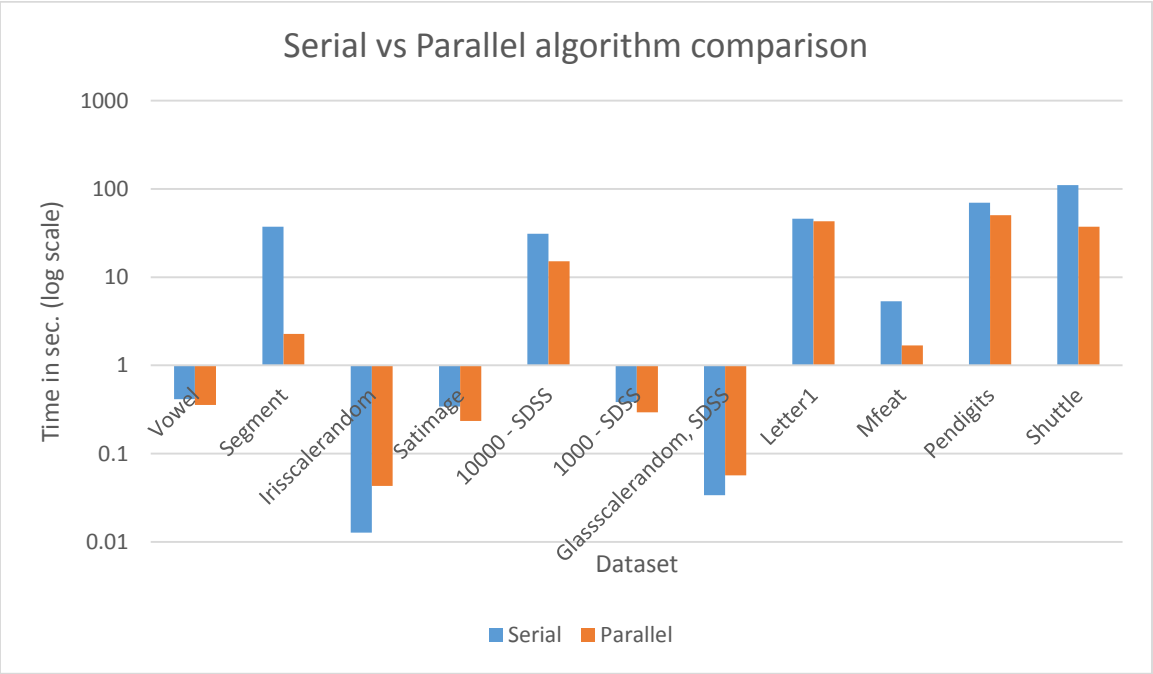
Fig 6.6



Above graphs were plotted for different Percentages of initially unlabeled data for each dataset, for each classifier – considering best choice of Threshold (as per the F-measure). As can be seen, F-measure of the model tends to fall as percentage of initially unlabeled data increases. This is intuitive. As the amount of labeled data in the initial data increases, the algorithm is not able to learn the pattern which exists in a representative sample of the dataset. Thus it fails in generalizing the pattern to the test set leading to a fall in F-measure.

Parallel Graphs

Fig 7



As can be seen from the graph, parallelizing the algorithm helps improve training time of the algorithm. Two of the most expensive steps in the algorithms are training SVM and Label propagation on the data. Doing these two in parallel reduces training time significantly (note that training time is in log scale)

5. Conclusion and Future work

The proposed approach uses SVM along with Label Propagation algorithm to yield a very high overall prediction quality. It can use a small amount of labeled data along with a large quantity of unlabeled data to yield a high F-measure on test data. It has a very small margin for error since it labels the unlabeled data using consent of both - SVM and Label Propagation. On testing both the algorithms on 12 different datasets we can conclude that the proposed approach performs much better than Label Propagation alone. It yields F-measure values which are almost twice as compared to Label Propagation. Further, we designed the parallel version of the approach and were able to decrease the training time significantly. In future, the parallel algorithm can be further enhanced to yield linear or super linear scale up. Also, more can be explored on using supervised approaches along with unsupervised approach to further improve prediction quality.

6. References

1. Castelli, V., & Cover, T. (1995). The exponential value of labeled samples. Pattern Recognition Letters, 16, 105–111.
2. Nigam, K., McCallum, A. K., Thrun, S., & Mitchell, T. (2000). Text classification from labeled and unlabeled documents using EM. Machine Learning, 39, 103–134.
3. Yarowsky, D. (1995). Unsupervised word sense disambiguation rivaling supervised methods. Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics (pp. 189–196).
4. Riloff, E., Wiebe, J., & Wilson, T. (2003). Learning subjective nouns using extraction pattern bootstrapping. Proceedings of the Seventh Conference on Natural Language Learning (CoNLL-2003).
5. Rosenberg, C., Hebert, M., & Schneiderman, H. (2005). Semi-supervised self-training of object detection models. Seventh IEEE Workshop on Applications of Computer Vision.
6. Zhu, Xiaojin, and Zoubin Ghahramani. Learning from labeled and unlabeled data with label propagation. Technical Report CMU-CALD-02-107, Carnegie Mellon University, 2002.
7. Label Propagation Through Linear Neighborhoods - Fei Wang, Changshui Zhang,2008 *machinelearning.wustl.edu/mlpapers/paper_files/icml2006_WangZ06.pdf*
8. Zhu, Xiaojin, John Lafferty, and Ronald Rosenfeld. *Semi-supervised learning with graphs*. Diss. Carnegie Mellon University, Language Technologies Institute, School of Computer Science, 2005
9. Cozman, Fabio Gagliardi, Ira Cohen, and Marcelo Cesar Cirelo. "Semi-supervised learning of mixture models." *ICML*. 2003
10. Cortes, Corinna, and Vladimir Vapnik. "Support-vector networks." *Machine learning* 20.3 (1995): 273-297
11. Scikit – *scikit-learn.org*