Leveraging Expectation Maximization for Identifying Claims in Low Resource Indian Languages

Rudra Dhar
Jadavpur University
West Bengal, India
rudradharrd@gmail.com

Dipankar Das
Jadavpur University
West Bengal, India
dipankar.dipnil2005@gmail.com

Abstract

Identification of the checkable claims is one of the important prior tasks while dealing with an infinite amount of data streaming from social web and the task becomes a compulsory one when we analyze them on behalf of a multilingual country like India that contains more than 1 billion people. In the present work, we describe our system which is made for detecting check-worthy claim sentences in resource scarce Indian languages (e.g., Bengali and Hindi). Firstly, we collected sentences from various sources in Bengali and Hindi and vectorized them with several NLP features. We labeled a small portion of them for check-worthy claims manually. However, in order to label the rest of data in a semi-supervised fashion, we employed the Expectation Maximization (EM) algorithm tuned with the Multivariate Gaussian Mixture Model (GMM) to assign weak labels. The optimal number of Gaussians in this algorithm is traced by using Logistic Regression. Furthermore, we used different ratios of manually labeled data and weakly labeled data to train our various machine learning models. We tabulated and plotted the performances of the models along with the stepwise decrement in proportion of manually labeled data. The experimental results were at par with our theoretical understanding, and we conclude that the weak labeling of check-worthy claim sentences in low resource languages with the EM algorithm has true potential.

1 Introduction

Misinformation has taken over the internet and it is now a well-recognized problem all over the world. Governments, news agencies, information security people all are trying to fight this menace with the helping hands from the researchers of social networks, natural language processing, data science and many more. With the exponential rise in information or news, making automated systems of fact-checking becomes a necessity. Researchers have made significant achievements in this field. But most of the research is in high resource languages like English. However, misinformation or fake news doesn't stop in the high resource language. Fake news is spreading far and wide in low resource languages as well, like in the Indian languages of Bengali and Hindi (Majumder and Das, 2020). A lot more research needs to be done in this arena and our present work we have contributed towards it.

It is a common practice in machine learning to use a semi supervised approach to weakly label data when labeled data is scarce (Xuan et al., 2010). On the other hand, filtering out check worthy claim sentences has been attempted by many teams as mentioned in (Hassan et al., 2015), (Dhar et al., 2019). While (Hassan et al., 2017) has made a generic and large system, (Anand et al., 2018) worked on twitter feeds. However, the problem of identifying claim sentences in Indian languages has not been attempted much and thus doesn't have a lot of labeled data. So here, we use the Expectation Maximization (EM) (Dellaert, 2003), a semi supervised algorithm to assign weakly labels to a lot of unlabeled data in Indian languages.

Suggested by (Hassan et al., 2015), automated fact checking is a two-fold task: 1) identification of check-worthy sentences, and 2) checking their trustworthiness based on some reliable sources. As checking the trustworthiness or veracity is a very resource intensive and computationally expensive job, identification or filtering out the check-worthy sentences becomes very important. In this work, we have used the semi supervised Expectation Maximization (EM) (Dellaert, 2003) algorithm to weakly label check-worthy sentences in the low resources Indian languages, Bengali and Hindi. Using semi supervised algorithms to label a lot of

Proceedings of the 18th International Conference on Natural Language Processing, pages 307–312
Silchar, India. December 16 - 19, 2021.

	# web pages crawled	# sentence (labeled)	# sentence (unlabeled)	Average length of sentence in Words
Bengali	400	1500	8568	11.40
Hindi	270	902	13537	13.70

Table 1: Statistics of our data.

unlabeled data is a common practice in Machine Learning and NLP. But, using EM or similar algorithms to weakly label check-worthy sentences has rarely been tried, and has certainly never been attempted in the context of Indian languages.

The rest of the Sections are organized as follows. Section 2 gives the approaches in developing the datasets while Section 3 and Section 4 describe the algorithms of Expectation Maximization (EM) and GMM, respectively. The experimental results are shown in Section 5 along with important observations. Finally, Section 6 concludes the draft.

2 Data Set Preparation

We prepared our own data for this work. We crawled sentences (mostly news) from various web sources and manually labeled some of them. We crawled Bengali sentences from 'ABP Ananda' (https://bengali.abplive.com/) and Hindi sentences from 'Abp News' (https://www.abplive.com/) and 'Aajtak' (https://aajtak.intoday.in/), as well as 'Twitter' (https://twitter.com/?lang=bn/hn).

We manually labeled a portion of the data (about 1500 for Bengali, and about 900 for Hindi). There were two classes. Class 1 refers to check-worthy claims, whereas class 0 refers to not-check-worthy sentences(that is, the rest of the sentences). Only the sentences, which is a fact checkable claim, with a clear context, are put in class 1. For example the sentence ('It is to be mentioned that in 2014, Rahul defeated Smriti Irani by 1 lakh 6 thousand votes in Amethi constituency.') is considered a check-worthy claim and hence would be labeled as class 1. Whereas the sentence ('The court summoned him on Thursday for this issue') is not considered a check-worthy claim and hence would be labeled as class 0.

We extracted tf-idf scores, unigram, bi-gram and part of speech tags, as features for both the languages. It has to be mentioned that the fitting of data in a Multivariate Gaussian Mixture Model is a very computationally expensive job. Experimentation by using all the thousands of features in our Gaussian Mixture Model (19907 for Bengali, and 27928 for Hindi) was not possible in our available hardware setup. Therefore, we used Principal Component Analysis (PCA) to reduce the dimensionality of the data to 1000. For Bengali we retained 92.9% data whereas in Hindi we retained 96.6% data.

3 EM - Semi-Supervised Algorithm

It is described in (Dellaert, 2003) that EM is an iterative optimization method to estimate some unknown parameters Θ, given measurement data U. However, we are not given some "hidden" nuisance variables J, which needs to be integrated out. In particular, we want to maximize the posterior probability of the parameters Θ given the data U, marginalizing over J:

$$\Theta^* = \arg\max_{\Theta} \sum_{J \in \tau^n} P(\Theta, J|U)$$

(Dempster et al., 1977) describes it from a statistical point of view in detail and (Little and Rubin, 2014) states its proof. The EM algorithm seeks to find the Maximum Likely Estimation of the marginal likelihood by iteratively applying the two steps namely **Expectation** and **Maximization**.

The EM algorithm needs a mixture model to represent the data. Here, we use 'Multivariate Gaussian Mixture Models' to represent the data. A Gaussian mixture model is a probabilistic model that assumes all the data points are generated from a mixture of a finite number of Gaussian distributions with unknown parameters. 'Multivariate Gaussian Mixture Models' are Gaussian Mixture Models in a multivariable or multidimensional vector space.

4 Experimental Setup

Firstly, we separated out the validation and the test sets from the manually labeled data with a train-validation-test split of 60%-20%-20% in ratio. Then, we made the training set using (strongly/human) labeled training data plus some of the unlabeled data while the validation and the test set consist of entirely labeled data. In each step of Expectation Maximization, we fit the training

No. of Gaussian	Class	Results on validation data					
		Bengali			Hindi		
		precision	recall	F1	precision	recall	F1
8	0	0.78	0.53	0.63	0.75	0.56	0.64
	1	0.38	0.66	0.48	0.40	0.60	0.48
16	0	**0.72**	**0.59**	**0.65**	0.75	0.53	0.62
	1	**0.41**	**0.56**	**0.47**	0.38	0.61	0.47
32	0	0.78	0.53	0.63	**0.78**	**0.65**	**0.71**
	1	0.38	0.66	0.48	**0.48**	**0.62**	**0.54**
64	0	0.78	0.53	0.63	0.81	0.53	0.64
	1	0.38	0.66	0.48	0.42	0.74	0.54

Table 2: Results on validation data given different number of Gaussian

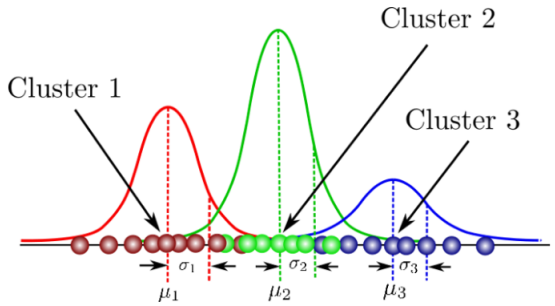


Figure 1: Single variable Gaussian mixture model with 3 Gaussian.

set to a Gaussian Mixture Model with a chosen number of Gaussians and assign the weak labels to the unlabeled data. Then, we measured the efficiency of the Gaussian Mixture Model by training a logistic regression component and evaluated on the validation set. We repeat this step and select the GMM which gives the best results on the validation set. At last, we report the performance of the selected GMM on the test set by modeling various Machine Learning models using human labeled data plus weakly labeled data. In addition, we have also explored this for various proportions of human (strongly) and weakly labeled data, and observed the performances with respect to the change in proportion. The details of each step are given in the following subsections as pseudo-code. (Note that we denote check-worthy claim sentences as class '1' and rest sentences as class '0') (shown in Figure 2).

4.1 Pseudo-code of the EM algorithm

- X_train = X_train_labeled + X_train_unlabled
- threshold = (number of claim in labeled data) / (number of labeled data)
  #threshold is the proportion of claims in labeled data. Later we will mark a set to be claim set, if it has higher proportion of claims than threshold
- choose 'gaussianNumber' as the number of Gaussians in our GMM.
- gModel = fit GMM with X_train
- get which training example is assigned to which Gaussian:
  yGaussian = gModel .predict(X_train)
- from yGaussian calculate number of strongly labeled '0' and '1' that fell under each Gaussian:
  for each Gaussian:
  - gy[0] = number of strongly labeled '0' in the Gaussian
  - gy[1] = number of strongly labeled '1' in the Gaussian
- calculate proportion of strongly labeled '1' in a Gaussian from gy
  for each Gaussian:
  - gRatio = gy[1] / (gy[0] + gy[1])
- put a label on each Gaussian:
  for each Gaussian:
  - if gRatio > threshold, then gaussianLabel = 1
  - else gaussianLabel = 0
- weakly label unlabeled data:
  y_train_weak_labled = 'gaussianLabel' of the Gaussian assigned to the sample.

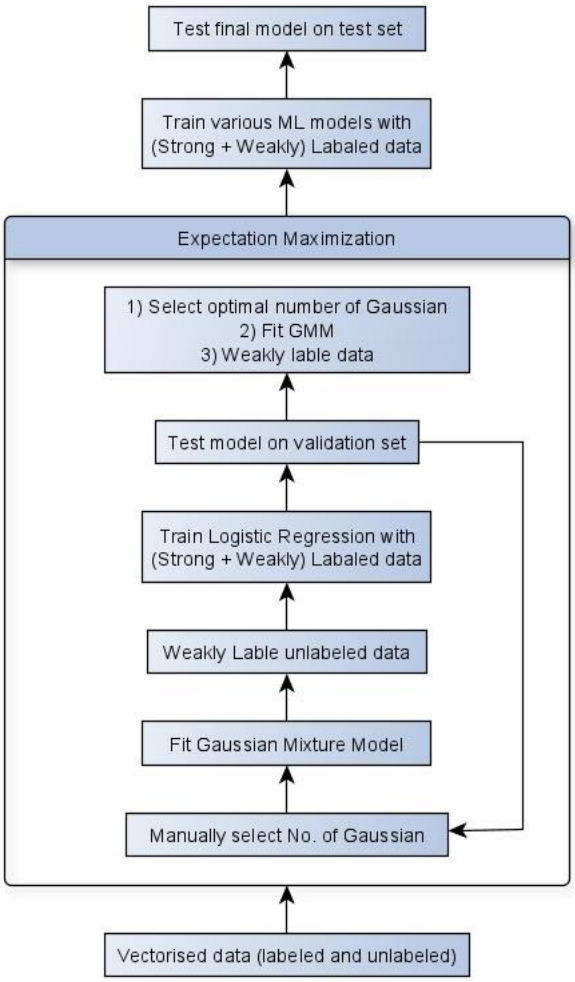


Figure 2: Architecture of the system including the EM algorithm.

- y_train = y_train_labled + y_train_weak_labled
- LModel = Logistic Regression model trained by(X_train, y_train)
- predicted_validation = predict value on validation set by LModel
- classification_report(true_validation, predicted_validation)

5 Experiments and Evaluation

We considered various Gaussian Mixture Models, with different numbers of Gaussian and selected the model which has the best performance on the validation data. We enlist the precision, recall, and the F1 score of the models on the validation data for both the classes and both the languages in Table 2. We manually observed these results and tried to determine the optimum number of Gaussians. We have noticed that the Gaussian mixture model with 16 numbers of Gaussians has the most promising results for Bengali while for Hindi a model with 32 numbers of Gaussian achieves the best result.

5.1 Training and Testing

After fixing the optimal Gaussian mixture model with the help of logistic regression, we train various machine learning models with human labeled data merged with semi-supervised weakly labeled data. However, we have fixed the number of human labeled data in the training set throughout the experiments. While we take various amounts of weakly labeled data and add them into the training set for identifying the impacts on utilizing variations in size of the weakly labeled data. We trained a logistic regression as well as Support Vector Machine (SVM) classifiers which are the classic machine learning models along with a neural network (Multi Layer Perceptron), and compared the results. For the neural networks, we used various MLP and chose the one giving the best results. In Table 3.1 and 3.2, we noted the performances of various classifiers on the validation set vs. size of weakly labeled data in the training set. (Note that class 1 is check-worthy claims, and class 0 is not-check-worthy claims) The number of weakly labeled data vs. the F1 score obtained in the test set is plotted in Figure 3, for both the languages and both the classes.

5.2 Observations

We observed that the performances of the models decrease with respect to the reduction in the proportion of human annotated data in the training set. This is also very much expected, as any machine learning algorithm's performance is supposed to deteriorate as the quality of its training data deteriorates.

This decrease in performance with increase in proportion of weakly labeled data is steeper in case of Hindi while for Bengali, it is more gentle. The gentle decline of performance with the increase in proportion of weakly labeled data for the Bengali dataset is an important observation. It shows that, annotating a lot of data with weak labels by employing a semi supervised algorithm, for detecting fact checkable claims is possible.

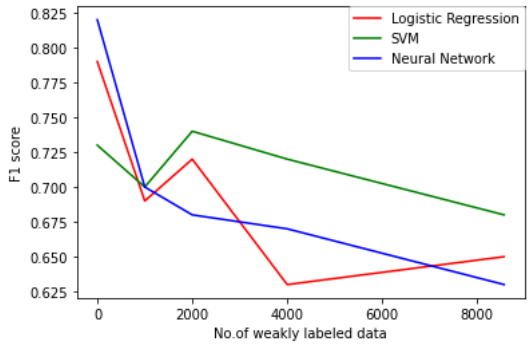
In the case of Bengali for both the classes, the performances of the classifiers like LR and SVM started improving while adding a weakly labeled data of size 1K to 2K while the neural networks

Size of Weakly Labeled Data	class	LR	SVM	Neural network	No. of sample in support
0	0 1	0.79 0.54	0.73 0.49	0.82 0.54	210 90
1000	0 1	0.69 0.45	0.70 0.40	0.70 0.49	210 90
2000	0 1	0.72 0.46	0.74 0.44	0.68 0.47	210 90
4000	0 1	0.63 0.38	0.72 0.44	0.67 0.46	210 90
8568	0 1	0.65 0.43	0.68 0.44	0.63 0.46	210 90

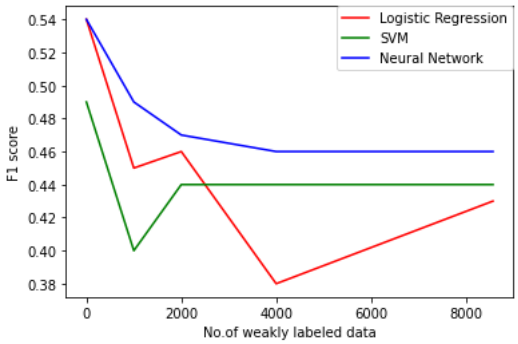
Table 3: F1 score for different classifiers with respect to different amounts of weakly labeled data in **Bengali**.

Size of Weakly Labeled Data	class	LR	SVM	Neural network	No. of sample in support
0	0 1	0.77 0.60	0.73 0.56	0.84 0.62	123 58
1000	0 1	0.82 0.61	0.74 0.55	0.83 0.68	123 58
2000	0 1	0.77 0.63	0.74 0.51	0.76 0.61	123 58
4000	0 1	0.71 0.60	0.71 0.47	0.73 0.54	123 58

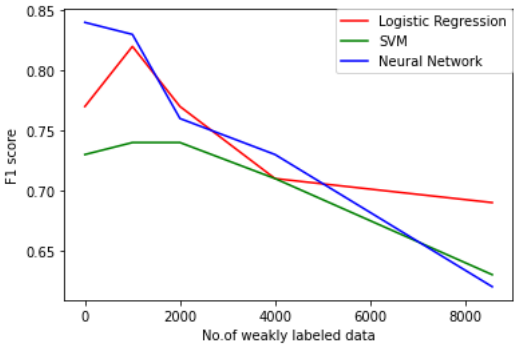
Table 4: F1 score for different classifiers with respect to different amounts of weakly labeled data in **Hindi**.



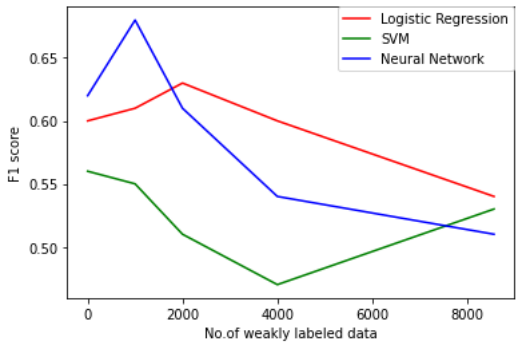
(a) Bengali class 0



(b) Bengali class 1



(c) Hindi class 0



(d) Hindi class 1

Figure 3: F1 score on test set vs No. of weakly labeled data in training set.

started degrading. In contrast, LR and neural networks started improving in case of Hindi data in the range of 500 to 1K except SVM. This is probably due to the fact that most machine learning algorithms perform better with more data. Thus, when a little amount of weakly labeled data is added to the training set, the performance of some of the models are improved. But, the performances of some of the models do not improve owing to the fact that the data we are adding is ultimately weakly labeled, and not by a human annotator.

6 Conclusion

In the present work, we have attempted to filter out check-worthy claims, which is the first step of fact-checking. Since there is a lot of information, and little intelligent labour available for manually tagging check-worthy sentences, we have used a semi-supervised algorithm to quickly tag a lot of check-worthy sentences by achieving a satisfied level of accuracy. We used the semi supervised Expectation Maximization algorithm with Gaussian Mixture Models, to weakly label the unlabeled data. We crawled a lot of data from the web and weakly labelled them using the Expectation Maximization algorithm. We trained some classification models with this weakly labeled data and observed the results on human annotated data. We find that the performance is very close to the performance by training with manually labeled data. Therefore, we can conclude that our semi supervised algorithm works well in the task of identifying fact checkable sentences.

We expect the efficiency will increase in future with more complicated models and more labeled data. For the Expectation maximization algorithm, computation power is also a concern. It is difficult to fit Gaussian Mixture models with high dimensions with currently available hardware. However we did proof that this approach works for identifying a lot of check-worthy claims with fair accuracy in Indian languages.

Acknowledgments

The present work is supported by the research project entitled "Claim Detection and Verification using Deep NLP: an Indian Perspective" funded by DRDO, Government of India.

References

Sarthak Anand, Rajat Gupta, Rajiv Ratn Shah, and Ponnurangam Kumaraguru. 2018. Fully automatic approach to identify factual or fact-checkable tweets. In *FIRE*.

Frank Dellaert. 2003. The expectation maximization algorithm.

Arthur P. Dempster, Nan M. Laird, and Donald B. Rubin. 1977. Maximum likelihood from incomplete data via the em - algorithm plus discussions on the paper.

Rudra Dhar, Subhabrata Dutta, and Dipankar Das. 2019. A hybrid model to rank sentences for check-worthiness. In *CLEF*.

Naeemul Hassan, Bill Adair, James Hamilton, Chengkai Li, Mark Tremayne, Jun Yang, and Cong Yu. 2015. The quest to automate fact-checking. *Proceedings of the 2015 Computation + Journalism Symposium*.

Naeemul Hassan, Anil Nayak, Vikas Sable, Chengkai Li, Mark Tremayne, Gensheng Zhang, Fatma Arslan, Josue Caraballo, Damian Jimenez, Siddhant Gawsane, Shohedul Hasan, Minumol Joseph, and Aaditya Kulkarni. 2017. Claimbuster: the first-ever end-to-end fact-checking system. *Proceedings of the VLDB Endowment*, 10:1945–1948.

Roderick Little and Donald Rubin. 2014. *Statistical Analysis with Missing Data, Second Edition*, pages 200–220.

Soumayan Bandhu Majumder and Dipankar Das. 2020. Detecting fake news spreaders on twitter using universal sentence encoder. In *CLEF*.

Jifeng Xuan, Jiang He, Zhilei Ren, Jun Yan, and Zhongxuan Luo. 2010. Automatic bug triage using semi-supervised text classification. pages 209–214.