Classifier Ensemble approach to Dependency Parsing

by

Silpa Kanneganti, vandan.mujadia , Dipti Misra Sharma

in

18th International Conference on Computational Linguistics and Intelligent Text Processing (*CICLing-2017*)

Budapest, Hungary

Report No: IIIT/TR/2017/-1

Centre for Language Technologies Research Centre
International Institute of Information Technology
Hyderabad - 500 032, INDIA
April 2017

Classifier Ensemble approach to Dependency Parsing

Silpa Kanneganti, Vandan Mujadia, Dipti M. Sharma

LTRC, International Institute of Information Technology,
Hyderabad

Abstract. In this paper we propose a neural network based classifier voting approach to dependency parsing using multiple classifiers as component systems in an ensemble and a neural network algorithm as an oracle. We show significant improvements over the best component systems for both transition-based and graph-based dependency parsing. We also investigate different weighting schemes for voting among individual classifiers in the ensemble. All our experiments were conducted on Hindi and Telugu language data but the approach is language-independent.

1 Introduction

Transition-based and graph-based parsing models are two of the most dominant approaches in dependency parsing. Transition-based parsers learn a model conditioned on parse history to score transitions from one parser state to the next. It employs a greedy algorithm, by taking the highest-scoring transition out of every parser state until a complete dependency graph is achieved. Graph-based parsers, learn a model to score dependency graphs for a given sentence by factoring them into their component arcs and perform parsing by searching for the highest-scoring graph.

Both models have been used to achieve state-of-the-art accuracy on dependency parsing for a wide range of languages [Buchholz and Marsi, 2006] [Nivre et al., 2007]. [Nivre and McDonald, 2008] proposes a method to integrate both the aforementioned models by letting the output of one guide the features for the other resulting in two stacked approaches, graph-based models guided by transition-based models and transition-based models guided by graph-based models. This approach is known to produce better accuracies than the isolated individual models.

While their approaches couldn't be more different from each other, both transition and graph-based models use single classifier based linear models to predict arcs or decisions for a given instance.

Recent studies suggest a classifier ensemble works better than a single classifier approach [Dietterich, 2000]. If a learning algorithm can be viewed as searching a space H of hypotheses to identify the best suited one in them, below are few issues faced by a single classifier approaches.

1. Statistical problems arise when the amount of training data available is too small compared to the size of the hypothesis space. Without enough data, the learning algorithm may find many different hypotheses in H that all give the same accuracy on the training data.
2. Computationally many learning algorithms work by performing some form of local search that may get stuck in local optima. Even in cases where there is enough training data it may still be very difficult computationally for the learning algorithm to find the best hypothesis.
3. Representationally speaking in most applications of machine learning, the true function f(classifier or regression function) cannot be represented by any of the hypotheses in H.

Hence the space H needs to be an effective space of hypotheses searched by the learning algorithm for a given training data set. Ensemble methods have the promise of reducing these three key shortcomings of standard learning algorithms .
In this paper as an alternative to the aforementioned single classifier approaches, we propose an ensemble method with several learners (multiple linear models) whose individual predictions are combined into a voting mechanism. Instead of using a majority based voting approach, to choose the best prediction, we train a neural network algorithm to predict the best classifier model given a feature vector. All the classifier models in the ensemble learn on the same set of train instance to produce predictions which are then validated against the gold standards values. These validations are then used as to train a neural network algorithm to learn the best possible performing classifier model for a given instance.

The key idea is to combine a number of classifiers such that the resulting system achieves higher classification accuracy and efficiency than the original single classifier models. Diversified multiple classifiers trained by different classifier parameters over the same train data prove to be more efficient [Kittler et al., 1998]. Hence, for each model we pick varied classifier models to get the best of each of them which helps reduce classifier related errors while confining to practical time constraints. Through experiments we show that our proposed ensemble model reports higher performances than the current state of the art single classifier models. All our experiments are performed on Hindi and Telugu language data.

2 Background and Related Work

Hindi is an Indo-Aryan language with richer morphology as compared to English. It exerts a relatively free word order with SOV being the default configuration. Due to the flexible word order, dependency representations are preferred over constituency for its syntactic analysis [Bharati and Sangal, 1993].

Telugu, a morphologically,syntactically complex language, is highly inflectional and agglutinative. It is a nominative-accusative language, with SOV as its default word order where the verbs exhibit a rich inflectional morphology. Hence it encodes various grammatical categories like tense, case, gender, number, person, negatives, imperatives etc. The dependency grammar formalism, used for both the languages is Computational Paninian Framework (CPG) [Begum et al., 2008][Bharati et al., 2009] . The data set we use for both the languages is from the ICON10 parsing contest [Husain et al., 2010] . For the purpose of this work we only deal with inter-chunk dependency trees.

Previous work on parser ensembling was based on models where, integration takes place at parsing time as well as at learning time, and requires at least three different base parsers. [Hall et al., 2007] combines six transition-based parsers and is so far the best performing system. [Nivre and McDonald, 2008] integrates the two parser models by allowing the output of one define features for the other. Feature-based integration performed by [McDonald et al., 2006] to substantial improvements in accuracy, lets a subset of the features for one model be derived from the output of a different model. In addition, feature-based integration has been used by [Taskar et al., 2005] , who train a discriminative word alignment model using features derived from the IBM models, and by [Florian et al., 2004] , who trained classifiers on auxiliary data to guide named entity classifiers. [Collins, 2000] perform perser re-ranking , where one parser produces a set of candidate parses and a second stage classifier chooses the most likely one. However, feature-based integration since is not explicitly constrained to any parse decisions that the guide model might make, is more efficient than parser re-ranking. [Nakagawa, 2007] and [Hall, 2007] try to add global features to to overcome the limited feature scope of graph-based models. [Titov and Henderson, 2007a] and [Titov and Henderson, 2007b] try to reduce error propagation , by performing beam search with globally normalized models for scoring transition sequences.

3 Proposed Model

In this section we describe in detail the proposed a neural network based classifier voting approach to dependency parsing.

3.1 Feature Representation

As explained in section 3, all the models essentially learn a scoring function s : X R, where the domain X is different for the two models. While for the transition-based model, X is the set of possible configuration-transition pairs (c, t), for the graph-based model, X is the set of possible dependency arcs (i, j, l); But in both cases, the input is represented by a k-dimensional feature vector f : X R k. For transition based approach, we use the feature models described in [Nivre et al., 2007]; For the graph-based models, we use the feature vectors defined in [Husain et al., 2010] and for integrated models we use the feature vectors from [Nivre and McDonald, 2008]

3.2 Ensemble Selection

Different kind of classifiers (i.e. independent, informed, diverse, etc) pick up different patterns in the data. Diversity, accuracy and run time are the three significant factors taken into account while picking the classifier ensemble. A necessary and sufficient condition for an ensemble of classifiers to be more accurate than any of its individual members is if the classifiers are accurate and diverse [Hansen and Salamon, 1990]. Two classifiers are diverse if they make different errors on new data points. In addition to this the selection and the number of classifiers used in the model are also confined by the total run time of the parser.

While the efficiency of the model keeps improving with the number of classifiers in the ensemble, keeping the aforementioned parameters in mind and various experiments, we picked 4 diverse classifiers each for both transition and graph based models without compromising on accuracy or run time of the parsers. while, Stochastic gradient descent and Linear ridge regression are common for both models, we use support vector machines (SVM) and Random Forest Tree classifier for transition-based models and Maximum Entropy and Decision Tree classifiers for graph-based models. We chose SVM for transition-based parsing for its proven high accuracy [Nivre et al., 2007], its theoretical guarantees to over-fitting, for higher dimension and non-linear data. We choose Maximum Entropy classifier for graph based parsing because of its efficiency with conditionally dependent data. We chpose Decision Tree Classifier because they easily handle feature interactions. They are non-parametric, so one doesnt have to worry about outliers or whether the data is linearly separable. We picked random forest tree for its is non-parametric and easily handle feature interactions with less concern for outliers.

We picked linear ridge regression because features co-relation is not necessary,the coefficients of linear transformation are normal distributed and the model is easily interpretable to analyze outputs, stochastic gradient descent because it is very efficient in discriminative learning of linear classifiers under convex loss functions. It is successful in sparse and large-scale learning because it is easier to scale sparse data. It has proven to be very successful in NLP tasks where large number of unique features are possible. We picked Multi-layer perceptron as the neural network classifier because of its ability to give more well structure blocks of layers to derive useful patterns from input data and its remarkable capability at deriving meanings (complex patterns) from imprecise data which may be too complex for other learning techniques.

3.3 Voting / Ensemble function

Given a feature vector, the output of each classifier is evaluated against the gold annotated output and corresponding binary values of 1 if the output is accurate and 0 if not is assigned to each vector. The resulting binary values are the mapped against their corresponding input feature vectors to create the training data for the Neural network. In order to handle redundancy and confusion, while creating the training data for the neural network, we ignore the instances where all the classifier predict the same value (0,1). This we believe creates a better learning model. We also added the corresponding accuracies of each of the classifiers as prior weights into the training instances for neural network. The neural nets are then trained on this data to produce the classifier method that works best for a given feature vector in the test data. Based on these predictions, we use the output of the respective model to predict the output.

Table 1 shows the example validation(nn_train data) data used to train the neural network model.

	classifier 1	classifier 2	classifier 3	classifier 4	nn prediction
feat 1	1	0	1	0	1
feat 2	1	0	1	1	3
feat 3	0	1	1	1	4
feat 4	1	0	1	1	1

Table 1. Accuracies

Given a set of training examples, S=(f(x_1),y_1),(f(x_2),y_2)....,(f(x_n),y_n) where f(x_i) is the feature vector and y_i their corresponding predictions, the proposed combined model C_n where n is the number of classifiers, produces a set of predictions P_i=y_{i1},y_{i2},....y_{in} for a given feature vector f(x_i) and are used to train the neural network (MLP-Multi layer perception) which learns to assigns scores to each of the classifiers based on their performance on the validation data. Given an test instance f(x_j), the resulting learning algorithm picks one of the classifiers in the ensemble c_n, as the possible model that works best for f(x_j). The data is divided into 3 parts:

- Train data: Used to train the feature vectors of the classifier models.
- NN_train data: Used to train the neural network(MLP) model.
- Test data: Used to test the model

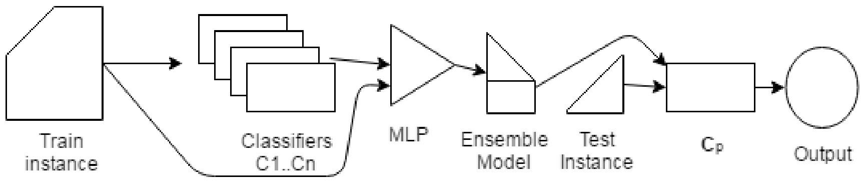

Fig. 1. Working of ensemble function C = Classifier, MLP = Multi Layer Perception

Figure 1 details the working of the proposed model, where the upper part shows the training of each of the classifiers in the ensemble on training instances and the predictions being passed to a Neural Network Model(MLP) as training data to help learn to decide the classifier model that works best for a given instance. The below part of the figure, shows the use of MLP to predict the best classifier(c_p) for a given test instance which in turn makes the final prediction.

We have used tensorflow software library [Abadi et al., 2015] for neural network implementation with 6 layered MLP with 4 hidden layers and they are consist of m,m/3,m/9,m/27 neurons respectively We have used RMSPropoptimizer[1] from tensorflow to minimize our objective function.

4 Experiments and Results

In this section, we present an experimental evaluation of the all the four aforementioned models. We conduct our experiments on Hindi and Telugu language data provided by [Husain et al., 2010] respectively. Due to data sparsity, 2 fold cross validation is done on the integrated models while 5 fold cross validation is done on transition and graph-based models. For transition-based and graph-based models,

– The 1452 annotated sentences in Telugu treebank data, are divided into 870 sentences of train data, 292 sentences of NN_train data and 290 of test data.
– Of the 19254 sentences, available in Hindi treebank data, 15404 sentences are used as train data, 3850 sentences are used as nn_train data and 3850 sentences as test data.
– For the integrated models, 400 sentences are used to train each of the graph-based and transition based models, 200 sentences each to train the respective neural network models and 252 sentences are used as test data.
– Of the 19254 sentences in HDTB data, 5000 sentences each are used to train the transition and graph based models, 3000 sentences each are used to train the neural network models and 3254 sentences are used as test data.

4.1 Transition Based Model

Transition based parsing systems use a model parameterized over transitions, such that every transition sequence from the designated initial configuration to some terminal configuration derives a valid dependency graph. The set of training instances for the learning problem are pairs (c, t) such that t is the correct transition out of c in the transition sequence that derives the correct dependency graph for some sentence x in the training set T. Each training instance (c, t) is represented by a feature vector f(c, t), where features are defined in terms of arbitrary properties of the configuration c, including the state of the stack, the input buffer and the partially built dependency graph.

Many features involve properties of the two target tokens that could be connected by an edge, the token on top of the stack and the first token in the input buffer. The full set of features used by the base model for Hindi is described in [Husain et al., 2010]. we use an implemented version of arc eager algorithm by [Bhat et al., 2016] and support vector machines as the base linear model to learn transition scores. In the proposed model, we use multiple discriminative methods to predict transitions over the same set of configurations. The validity vectors of resulting transitions are then fed to the neural

[1] http://www.cs.toronto.edu/˜tijmen/csc321/slides/lecture_slides_lec6.pdf

network algorithm which inturn predicts the transitions of the test feature configurations. The data is divided into 3 parts, data to train the classifier models, data to train the neural network model, test data.

	Hindi			Telugu		
Classifier Model	LAS	UAS	LS	LAS	UAS	LS
Support Vector Machines(Base model)	**78.29%**	**85.36%**	**80.02%**	**62.17%**	**70.35%**	**64.55%**
Random Forest Tree	77.58%	85.03%	79.71%	61.50%	69.71%	63.74%
Stochastic Gradient Descent	76.64%	84.56%	78.5%	61.71%	69.97%	63.23%
Linear Regression(RidgeCV)	76.14%	84.22%	78.63%	62.10%	70.25%	64.58%
Ensemble Model	**79.04%**	86.16%	81.63%	**62.66%**	**71.03%**	**64.98%**

Table 2. Accuracies

Table 2 shows the accuracies[2] of best accuracies for each of the classifiers and the ensemble model.

4.2 Graph based Model

Graph-based dependency parsers parameterize a model over smaller substructures in order to search the space of valid dependency graphs and produce the most likely one. The simplest parameterization is the arc-factored model that defines a real-valued score function for arcs s(i, j, l) and further defines the score of a dependency graph as the sum of the score of all the arcs it contains. The specific graph-based model studied in this work is that presented by [McDonald et al., 2005], which factors scores over pairs of arcs (instead of just single arcs) and uses near exhaustive search for unlabeled parsing coupled with a separate classifier to label each arc. We use Maxent as the base line classifier with the settings suggested in [Husain et al., 2010]. Since the ensemble model is used only to predict the labels of predicted trees, there is no change the Unlabeled Attachment score(UAS) and hence the Labeled Attachment score(LAS) and Labeled Accuracy(LA) are the same.

Table 3 shows the best accuracies for each of the classifiers as well as the ensemble model. The **UAS** on graph based models is same for all the models. This can be attributed to the fact that these classifier models are used only to predict the labels while the trees predicted my the parser remain constant for all of them.

[2] LAS: Labeled Accuracy Score; UAS: Unlabled Accuracy Score; LS: Labeled Accuracy score

Classifier Model	Hindi			Telugu		
	LAS	UAS	LS	LAS	UAS	LS
Maxent(Base model)	**78.63%**	**85.94%**	**78.63%**	**62.34%**	**70.93%**	**62.34%**
Decision Tree	77.34%	85.94%	77.34%	61.90%	70.93%	61.90%
Stochastic Gradient Descent	77.05%	85.94%	77.05%	61.21%	70.93%	61.21%
Linear Regression(RidgeCV)	78.64%	85.94%	78.64%	62.82%	70.93%	62.82%
Ensemble Model	**79.42%**	**85.94%**	**79.42%**	**62.92%**	**70.93%**	**62.92%**

Table 3. Accuracies

4.3 Guided Transition Based Model

Guided transition-based models follow the same perspective as transition-based models but for modified feature configurations. The basic training instances of a transition-based model are extended to add the features predicted by the graph-based parser for a given sentence. We therefore use the same classifier settings and approach we used for the transition-based parsing model discussed in section 4.1 . The data is divided into 5 parts. Data used to train the base model, the guide model, neural network models of the base and guide models as well as the test data run by the base model. The full set of features used by the guided transition-based model are described in [Nivre and McDonald, 2008].

Classifier Model	Hindi			Telugu		
	LAS	UAS	LS	LAS	UAS	LS
Support Vector Machines(Base model)	**71.09%**	**79.15%**	**73.3%**	**55.59%**	**63.81%**	**57.24%**
Random Forest Tree	70.46%	78.92%	72.71%	54.02%	62.71%	56.19%
Stochastic Gradient Descent	70.53%	78.02%	72.18%	54.71%	62.97%	56.23%
Linear Regression(RidgeCV)	71.21%	79.43%	73.65%	55.10%	63.25%	57.58%
Ensemble Model	**72.14%**	**80.21%**	**74.13%**	**56.16%**	**64.03%**	**58.98%**

Table 4. Accuracies

Table 4 shows the best accuracies for each of the classifiers as well as the ensemble model.

4.4 Guided Graph Based Model

Guided graph-based models follow the same perspective as graph-based models but for modified feature vectors. The basic training instances of a graph-based model are extended to add the features predicted by the transition-based parser for a given sentence. We therefore use the same classifier settings and approach we used for the graph-based parsing model discussed in section 4.2 . The data is used as discussed in section 4.3. The full set of features used by the guided graph-based model are discussed in [Nivre and McDonald, 2008]. Since the ensemble model is used only to predict the labels of predicted trees, there is no change the Unlabeled Attachment score(UAS). Similar to the graph-based models, since the ensemble model is used only to predict the labels of already parsed trees, there is no change the Unlabeled Attachment score(UAS) and hence the Labeled Attachment score(LAS) and Labeled Accuracy(LA) are the same.

	Hindi			Telugu		
Classifier Model	LAS	UAS	LS	LAS	UAS	LS
Maxent(Base model)	**71.56%**	**79.86%**	**71.56%**	**55.83%**	**64.25%**	**55.83%**
Decision Tree	70.44%	79.86%	70.44%	54.71%	64.25%	54.71%
Stochastic Gradient Descent	70.26%	79.86%	70.26%	54.38%	64.25%	54.38%
Linear Regression(RidgeCV)	71.94%	79.86%	71.94%	56.13%	64.25%	56.13%
Ensemble Model	**72.44%**	**79.86%**	**72.44%**	**56.48%**	**64.25%**	**56.48%**

Table 5. Accuracies

Table 5 shows the best accuracies for each of the classifiers and the ensemble model. Similar to the graph based models in section 4.2, given that the models are only predicting dependency labels, the tree structures between them remain the same. Hence **UAS** is common among all of them.

5 Observations

One of the main assumptions in using diverse classifiers while choosing the ensemble is to learn to handle the cases where output generated by those models may differ. The notion that agreement between the models, an indication of correctness might create confusion for the neural network to learn in this scenario worked in our favor. Experiments with training the neural network with only the instances where atleast 2 classifiers disagree proved to be more helpful than all the samples in case of Hindi data for all the models. Telugu data on the other hand shows improvements only for transition and graph based model. For integrated models, due to scarcity of already split data in Telugu, no training instance could be ignored.

In all the models, assigning prior weights to classifiers based on their individual performances has proven to get the best results for both the languages. We also observed that a random initial weight settings to classifier voting for neural networks performed surprisingly well, although not better than performance based weight assignment. It is interesting to note that although the baseline classifier SVM is giving the best results compared to the rest in transition-based models, Liner regression model is performing better than the baseline model Maxent for graph-based models. Analysis on the results suggest that the performance of ensemble models are heavily dependent on the size of the data sets. In the integrated models a huge chunk of the data goes into training the guide model as well as the corresponding neural network models. Hence the drop in baseline as well as ensemble accuracies compared to the simple parsers. Also the improvement accuracies for Hindi data is higher than Telugu on all the models. We also noticed that performance enhancement for an ensemble comes with the first few classifiers combined peaking at a threshold even with increase in ensemble sizes. In conclusion, as a general technique with the right pick of classifier models and the weight assignment, ensemble models may produce larger gains in accuracy. Figures 2 and 3 show the plotted statistics of transition and graph based models repectively.

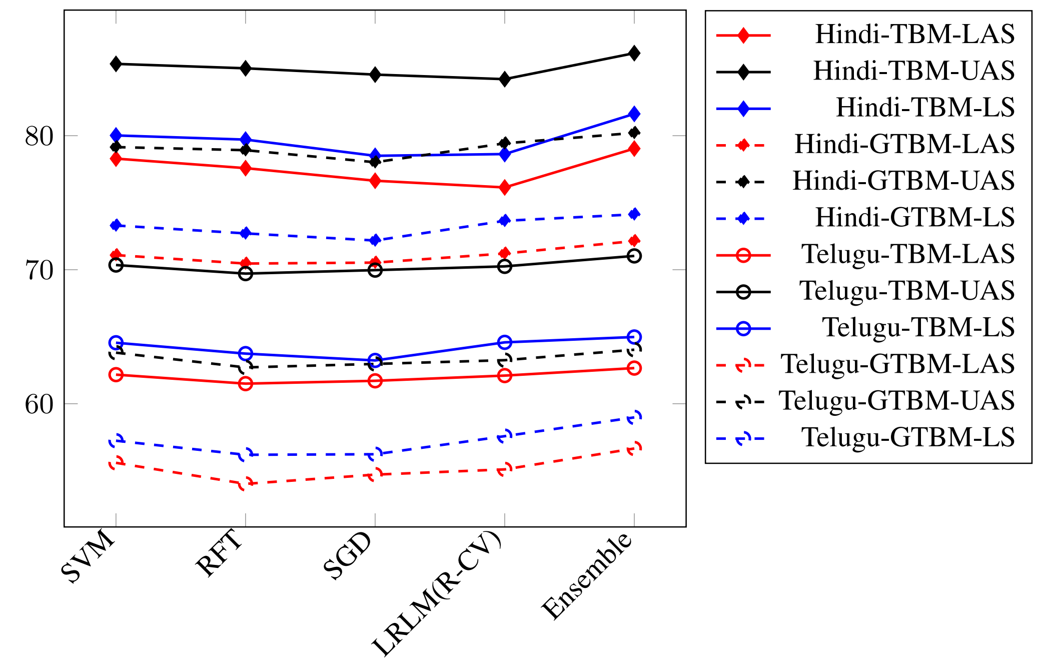

Fig. 2. accuracy graph, TBM=Transition based model; GTBM=Guided Transition based model, LAS=Label attachment score; UAS=Unlabeled attachment score; LS=Label accuracy score

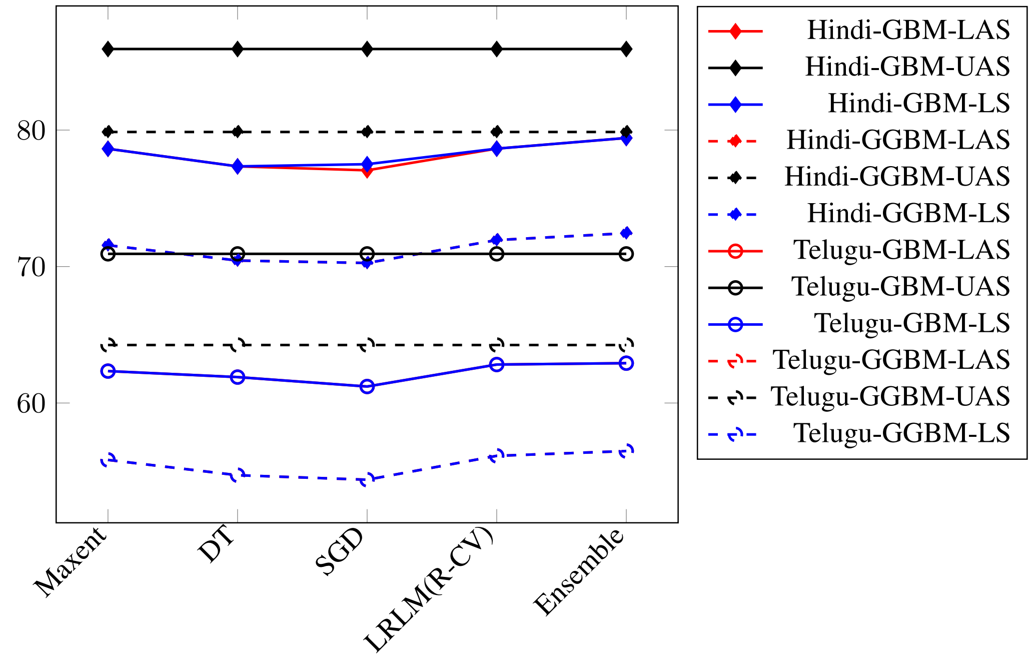

Fig. 3. accuracy graph, GBM=Graph based model; GGBM=Guided Graph based model, LAS=Label attachment score; UAS=Unlabeled attachment score; LS=Label accuracy score

6 Conclusion

The idea of ensemble modeling in parsing is not new. While work on parser ensembling usually makes use of a voting strategy of some kind in order to derive a single prediction for an output, we deal with classifier ensembles within a parser. Our work differs from that of the parser ensembles in that integration is done during learning of the parse models. In addition to the simple parse models we also show our technique improves the integrated models described in [Nivre and McDonald, 2008]. We also show how choosing diverse classifiers help in expanding to cover more hyptheses space resulting in reduced classifier based errors.

References

[Abadi et al., 2015] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

[Begum et al., 2008] Begum, R., Husain, S., Dhwaj, A., Sharma, D. M., Bai, L., and Sangal, R. (2008). Dependency annotation scheme for indian languages. In *IJCNLP*, pages 721–726. Citeseer.

[Bharati et al., 2009] Bharati, A., Husain, S., Misra, D., and Sangal, R. (2009). Two stage constraint based hybrid approach to free word order language dependency parsing. In *Proceedings of the 11th International Conference on Parsing Technologies*, pages 77–80. Association for Computational Linguistics.

[Bharati and Sangal, 1993] Bharati, A. and Sangal, R. (1993). Parsing free word order languages in the paninian framework. In *Proceedings of the 31st annual meeting on Association for Computational Linguistics*, pages 105–111. Association for Computational Linguistics.

[Bhat et al., 2016] Bhat, R. A., Bhat, I. A., and Sharam, D. M. (2016). Improving dependency parsing of hindi and urdu by modeling syntactically relevant phenomena. *ACM Transactions on Asian and Low-Resource Language Information Processing (under review)*.

[Buchholz and Marsi, 2006] Buchholz, S. and Marsi, E. (2006). Conll-x shared task on multilingual dependency parsing. In *Proceedings of the Tenth Conference on Computational Natural Language Learning*, pages 149–164. Association for Computational Linguistics.

[Collins, 2000] Collins, M. (2000). Discriminative reranking for natural language parsing. In *Proceedings of the Seventeenth International Conference on Machine Learning*, ICML '00, pages 175–182, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

[Dietterich, 2000] Dietterich, T. G. (2000). Ensemble methods in machine learning. In *International workshop on multiple classifier systems*, pages 1–15. Springer.

[Florian et al., 2004] Florian, R., Hassan, H., Ittycheriah, A., Jing, H., Kambhatla, N., Luo, X., Nicolov, H., and Roukos, S. (2004). A statistical model for multilingual entity detection and tracking. Technical report, DTIC Document.

[Hall et al., 2007] Hall, J., Nivre, J., and Nilsson, J. (2007). A hybrid constituency-dependency parser for swedish. In *Proceedings of NODALIDA*, pages 284–287.

[Hall, 2007] Hall, K. (2007). K-best spanning tree parsing. In *Annual Meeting-Association for Computational Linguistics*, volume 45, page 392.

[Hansen and Salamon, 1990] Hansen, L. K. and Salamon, P. (1990). Neural network ensembles. *IEEE transactions on pattern analysis and machine intelligence*, 12:993–1001.

[Husain et al., 2010] Husain, S., Mannem, P., Ambati, B., and Gadde, P. (2010). The icon-2010 tools contest on indian language dependency parsing. *Proceedings of ICON-2010 Tools Contest on Indian Language Dependency Parsing, ICON*, 10:1–8.

[Kittler et al., 1998] Kittler, J., Hatef, M., Duin, R. P., and Matas, J. (1998). On combining classifiers. *IEEE transactions on pattern analysis and machine intelligence*, 20(3):226–239.

[McDonald et al., 2005] McDonald, R., Crammer, K., and Pereira, F. (2005). Online large-margin training of dependency parsers. In *Proceedings of the 43rd annual meeting on association for computational linguistics*, pages 91–98. Association for Computational Linguistics.

[McDonald et al., 2006] McDonald, R., Lerman, K., and Pereira, F. (2006). Multilingual dependency analysis with a two-stage discriminative parser. In *Proceedings of the Tenth Conference on Computational Natural Language Learning*, pages 216–220. Association for Computational Linguistics.

[Nakagawa, 2007] Nakagawa, T. (2007). Multilingual dependency parsing using global features. In *EMNLP-CoNLL*, pages 952–956. Citeseer.

[Nivre et al., 2007] Nivre, J., Hall, J., Nilsson, J., Chanev, A., Eryigit, G., Kübler, S., Marinov, S., and Marsi, E. (2007). Maltparser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13(02):95–135.

[Nivre and McDonald, 2008] Nivre, J. and McDonald, R. T. (2008). Integrating graph-based and transition-based dependency parsers. In *ACL*, pages 950–958.

[Taskar et al., 2005] Taskar, B., Lacoste-Julien, S., and Klein, D. (2005). A discriminative matching approach to word alignment. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 73–80. Association for Computational Linguistics.

[Titov and Henderson, 2007a] Titov, I. and Henderson, J. (2007a). Constituent parsing with incremental sigmoid belief networks. In *ANNUAL MEETING-ASSOCIATION FOR COMPUTATIONAL LINGUISTICS*, volume 45, page 632.

[Titov and Henderson, 2007b] Titov, I. and Henderson, J. (2007b). Fast and robust multilingual dependency parsing with a generative latent variable model. In *EMNLP-CoNLL*, pages 947–951.