A Dynamic Confusion Score for Dependency Arc Labels

Sambhav Jain and **Bhasha Agrawal**
Language Technologies Research Center
IIIT-Hyderabad, India
{sambhav.jain, bhasha.agrawal}@research.iiit.ac.in

Abstract

In this paper we propose an approach to dynamically compute a *confusion score* for dependency arc labels, in typed dependency parsing framework. This score accompanies the parsed output and aims to administer an informed account of *parse correctness*, detailed down to each edge of the parse. The methodology explores the confusion encountered by the oracle of a data driven parser, in predicting an arc label. We support our hypothesis by empirically illustrating, for 20 languages, that the labels with a high confusion score are notably the predominant parsing errors.

1 Introduction

Recently, a major research drive has been towards building data driven dependency parsers for various languages. Shared tasks like CoNLL-X and CoNLL 2007 have acted as development and testing grounds for various efforts in the field. The majority of the emerged systems follow either *graph based paradigm* (McDonald et al., 2005; McDonald and Pereira, 2006) eg. MST Parser (McDonald et al., 2005) or *transition based paradigm* (Yamada and Matsumoto, 2003; Nivre and Scholz, 2004) eg. MaltParser(Nivre et al., 2007).

A time complexity relatively lower than their earlier counterparts makes the above parsers apt for use by real time NLP applications like Machine Translation (Galley and Manning, 2009). However, Popel et al.(2011) in the context of MT pointed out that an incorrect parse can hurt the accuracy of output. Thus, correctness of individual parses becomes a key factor in such a setup.

This calls for measures which can indicate upfront the quality of each individual output. A relevant work in this direction is by Mejer and Crammer (2012). They proposed methods to estimate *confidence of correctness* of predicted parse in a graph based parsing scenario using MSTParser. Graph-based and transition-based parsers exhibit two very distinctive approaches towards parsing, each having its own strengths and limitations (McDonald and Nivre, 2007; Zhang and Clark, 2008). The diversity in the two techniques motivated us to explore and formulate a similar measure in transition based paradigm. We choose MaltParser[1] (Nivre et al., 2007), which produces a parse tree using a *shift-reduce* based transition algorithm, to work with. It uses *SVM* to train an oracle to predict parsing action. The measures proposed by Mejer and Crammer (2012) can not be straight off applied in transition based parsers, as unlike the graph based approach they commit local operations and thus can not directly produce globally optimum k-best parses.

We propose computing an entropy based label confusion score, dynamically computed and assigned while parsing (the computational details are presented in section 2 of this paper). Since entropy measures the uncertainty in a random variable, we prefer to call the measure, in our approach, *confusion score*. This measure aims to give a more informed picture of the parsed output.

We integrated our approach on top of the current functionality of MaltParser, adjusting it to accredit a confusion score with each arc label predicted in the output. Figure 1 depicts a typical parse from our proposed system where each arc label has been designated a confusion score.

The measure can also be utilized to flag potential *incorrectly parsed edges* which later, can either be manually corrected or altogether discarded (to fall back on lower level but more accurate features). We empirically illustrate in section 4, that such a score can be effectively used for automatic error detection in parsed outputs and guide manual

[1]MaltParser version 1.7 from *http://www.maltparser.org*

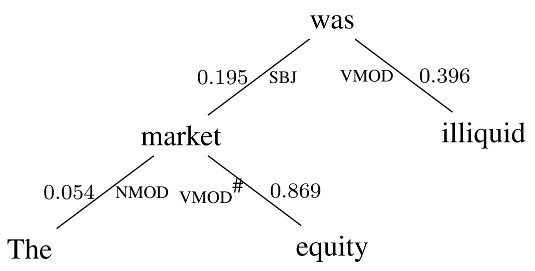

Figure 1: A dependency parse tree with edge confusion score. '#' represents incorrect arc label.

correction.

2 Dynamic Confusion Score

MaltParser outputs a single best parse by greedily choosing parsing actions advocated by an oracle trained on the training data. In a typed dependency framework, the parser performs two distinct kinds of actions; attachment of vertices and assigning arc labels to edges. MaltParser provides a choice[2] to train separate oracles for these two kinds or a single oracle that jointly predicts attachment and arc label.

In case of separate oracles for attachment and label prediction, the parser queries the attachment oracle until a *Left Arc* or *Right Arc* action is predicted. The label oracle is then queried for an arc label in the given context. There is further a provision for having a separate oracle for *Left Arc* and *Right Arc* label prediction. Nevertheless, an oracle is always queried for a parsing action against a given context.

2.1 Uncertainty in Label Prediction

The problem of predicting arc label correctness can mathematically be posed as follows: For random variables X and Y representing context and parsing-actions respectively; an oracle ϕ is defined such that $\phi : X \longrightarrow Y$. Here, Y is always a closed set, comprising permissible parser actions. The uncertainty in predicting Y to one of its possible values $y_1, ..., y_n$ can be attributed as the confusion, the oracle has in prediction. It is known that entropy is a measure of the uncertainty in a random variable (Ihara, 1993). Thus, this uncertainty can be quantitatively determined by *entropy(H)* calculated by the following formula

$$H(Y/X) = -\sum_{i=1}^{n} p(y_i/X) \log p(y_i/X) \quad (1)$$

[2]http://www.maltparser.org/userguide.html#predstrate

were $p(y_i/X)$ is the posterior probability of y_i being predicted as the parsing action in the given context X. The higher the entropy, the more uncertain the oracle is about the prediction.

However, there is no readily available provision indicating the magnitude of confusion the oracle encounters during prediction. The rest of this section presents a sequential account of our approach.

2.2 SVM based Oracle

The oracle discussed earlier, is a multiclass classifier which predicts a transition action based on the context. MaltParser employs Support Vector Machine (Cortes and Vapnik, 1995) for classification and provides an option between LIBSVM (Chang and Lin, 2011) and LIBLINEAR (Fan et al., 2008) to build the classifier(s). Both implement *"one-vs-one multi-class classification"* method which incorporates ${}^nC_2$ binary models (n denotes number of classes), one for each distinct pair of classes. Prediction is done by voting among these binary classifiers and the class with maximum votes is emitted as decision class. This method does not exert any sort of probabilities and in our scenario we seek posterior probability estimates of the classes.

2.3 Posterior Probabilities

Platt et al. (1999) showed that posterior probabilities can be estimated in SVM by training the parameters of an additional sigmoid function to map the SVM output into probabilities. Later, Wu et al. (2004) extended the idea for multiclass probability estimates by combining pairwise class probabilities. In our work, we utilize the second method (proposed in Wu et al. (2004)), which suggests the following optimization formula:-

$$\min_p \sum_{i=1}^{k} \sum_{j:j\neq i} (r_{ji}p_i - r_{ij}p_j)^2$$

$$\text{subject to } \sum_{i=1}^{k} p_i = 1, p_i \geq 0, \forall i$$

$where,$

$k =$ total classes,

$r_{ij} =$ probability of i in binary classifier with classes i & j,

$p_i =$ probability of i in multiclass estimation

The solution to above optimization, furnishes multiclass estimation of probability for each class.

2.4 Entropy as Confusion Measure

Now, with the available class posterior probabilities, entropy based confusion measure is computed using equation 1. The base of the log is the number of label classes which ensures the entropy to be in the range of 0 to 1. Confusion score for arc label would require querying the arc label oracle, and thus MaltParser must be configured to deliver separate oracles for attachment and arc label in the training phase.

3 Extendibility of the Confusion Score

This section presents the possible extensions and constraints of the proposed score.

3.1 Extension to Full Parse Confusion

The confusion score in the proposed approach is calculated separately for each arc label. Calculation of a confusion score for a full parse can be done by taking an average over the edges in the parse. Other measures like average of worst k labels, score of worst label itself, etc. can also be adopted. This enables visualization of the confusion for the complete parse tree. This can be apt in the scenario of a large collection of parse trees, such as treebanks and also for applications like Active Learning (Tang et al., 2002; Hwa, 2004).

3.2 Extendibility to Other Algorithms

Since our method executes at the oracle level, it is independent of the algorithmic choice used in the parser. Also, pseudo projective transformation (Nivre and Nilsson, 2005) too is an extrinsic process, so non-projectivity does not perturb our approach.

3.3 Extendibility for Attachments

The approach, at first, may seem extendable to attachments also since it would require querying the attachment oracle for attachment confusion. However it is not extendable to edge correctness prediction. The restricting factor being the presence of non-labeling parser action i.e. $Shift$ and $Reduce$. Since these transitions are not decomposed over the tree edges, the oracle confusion associated with them can not be delineated to any edges. For example, at a given point in the parsing process, assuming the arc-standard system (but the same holds for other algorithms also), the oracle will need to decide if it should perform a $Left$, $Right$ or $Shift$ action. This decision will influence not only the single edges added by the left or right operation, but also other, future edges. It should be noticed that the Shift action will not add any edge, but will have a very complex effect on the set of edges that could or could not be added in the future. Thus, the entropy of this particular decision cannot be attached to any specific final edge and hence the methodology can not be extendend to arc-attachments.

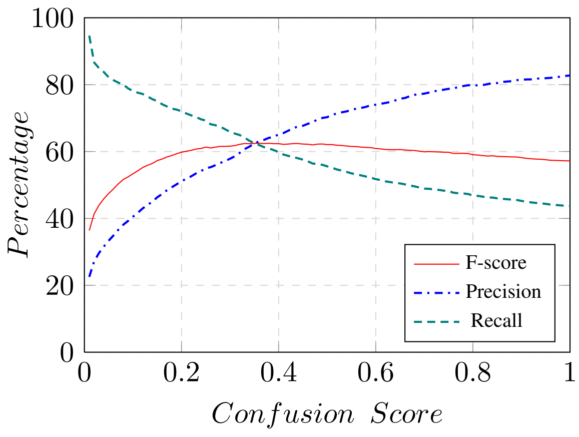

Figure 2: *Precision, recall and f-score for various values of confusion score on 'Hungarian' development set.*

4 Error Detection in Parser Output

In this section, we empirically illustrate the efficacy of our proposed measure in automatic error detection and guiding manual error correction.

4.1 Automatic Error Detection

Automatic error detection aims to efficiently determine and flag incorrectly predicted edges. The edges exhibiting a high confusion score are also highly probable to be incorrect, as the oracle is uncertain in its decision. Using this insight, an edge-label is flagged as potential error if its confusion score is above a pre-calculated threshold(θ).

In this task we have focused on the arc label correctness, i.e. we flag the edges which have a high probability for an incorrect arc label.

4.2 Data and Experimental Setup

We conducted experiments on 20 languages, using data from CoNLL-X (Buchholz and Marsi, 2006), CoNLL 2007 (Nilsson et al., 2007) and MTPIL COLING 2012 (Sharma et al., 2012) shared tasks on dependency parsing. We carried out experiments on the systems proposed in Nivre et al.

Language	Threshold (θ)	F-score (%)	Precision (%)	Recall (%)	EDI 1% edges(%)	EDI 5% edges(%)	EDI 10% edges(%)
Arabic#	0.14	50.71	41.39	65.45	4.43	20.98	36.68
Basque#	0.16	51.74	46.62	58.13	2.57	9.58	24.41
Catalan#	0.11	48.67	48.54	48.79	7.63	26.53	44.79
Chinese#	0.09	41.35	41.68	41.04	5.70	20.02	36.00
Czech#	0.08	51.61	47.85	56.02	4.20	18.82	35.92
English#	0.09	47.71	57.78	40.63	8.54	33.89	44.58
Greek#	0.12	54.47	44.76	69.54	4.84	20.28	35.89
Hungarian#	0.36	61.80	69.64	55.54	6.23	31.14	53.49
Italian#	0.15	43.48	39.43	48.45	2.57	19.73	40.37
Turkish#	0.14	53.58	43.38	70.05	3.99	21.90	40.82
Hindi♠	0.16	49.80	43.57	58.11	4.86	20.48	35.09
Bulgarian¶	0.12	47.52	40.45	57.60	7.85	34.88	55.63
Danish¶	0.12	49.77	41.85	61.41	6.26	30.39	50.32
Dutch¶	0.11	50.29	47.46	53.47	2.63	15.39	34.25
German¶	0.09	44.44	37.33	54.91	4.37	23.62	45.64
Japanese¶	0.09	41.43	29.12	71.75	4.90	24.49	**57.59**
Portuguese¶	0.11	48.13	44.61	52.25	**10.43**	**35.23**	54.51
Slovene¶	0.14	54.29	44.84	68.78	5.72	19.38	34.15
Spanish¶	0.09	40.00	31.10	56.03	7.43	29.21	46.33
Swedish¶	0.13	48.47	42.36	56.63	5.03	24.04	42.37
Average	-	*48.96*	*44.19*	*57.23*	*5.51*	*24.00*	*42.44*

Table 1: Language wise results for automatic error detection task. EDI x% edges= Error detected on inspecting top x% of total edges. Data Source:- ¶:CoNLL-X, #:CoNLL 2007, ♠:MTPIL COLING 2012

(2006), Hall et al. (2007) and Singla et al. (2012), which are individually, the best performing MaltParser based systems, in the respective shared tasks. All the results reported here are on the official test sets.

4.3 Identifying Optimum Threshold(θ)

Threshold(θ) is a crucial parameter in the experimental setup. An optimum θ is chosen by making use of the development set. We iteratively increase candidate values for θ, from minimum to maximum possible value of confusion score, with an adequate interval. Corresponding to each of these values, the incorrect edges are flagged and $precision$, $recall$ & $F\text{-}score$ (Manning et al., 2008) are calculated. The value asserting the maximum $F\text{-}score$ is chosen as the final θ. Here for simplicity, we have used balanced $F\text{-}score$, i.e. $F_1\text{-}score$. However, as per the application and available resources, a relevant F_β can be chosen to maximize the yield on the input effort.

$$F_\beta = (1+\beta^2) \times \frac{precision \times recall}{(\beta^2 \times precision) + recall}$$

Figure 2 depicts $precision$, $recall$ and $F_1\text{-}score$ corresponding to each candidate value of θ for Hungarian development data. The maximum $F_1\text{-}score$ is attained at 0.36 which is thus taken as the final θ for Hungarian.

Since, CoNLL-X and CoNLL 2007 datasets do not provide development sets, we hold out random 10% sentences of a training set as development data. The remaining training data is utilized to train a parser and identify an optimum threshold on the development set, as explained earlier. However, the final training is performed on the entire training data and evaluated on the test set.

4.4 Results and Discussion

Table 1 exhibits the results obtained for automatic error identification. An average $F_1\text{-}score$ of 48.96%, $precision$ of 44.19% and $recall$ of 57.23% is obtained over 20 languages in the task.

To efficiently capture the efficacy of our approach, another metric is presented (columns 6-8) which corresponds to the percentage of errors detected by inspecting top 1%, 5% and 10% of total edges. The metric gives a more precise picture of the effort required to correct errors.

Our experiments indicate that on average 42.44% errors can be detected by just inspecting top 10% of total edges. This portrays that the effort required here is one fourth as compared to that in conventional sequential correction. On inspecting top 5% and 1% of all edges, 24.00% and 5.51% errors can be detected. Best results are obtained for Japanese and Portuguese where 57.59% of errors are detected by merely inspecting 10% of total edges for Japanese, while 35.23% and 10.43% errors are detected on inspecting 5% and 1% edges respectively for Portuguese.

A comparison with Mejer and Crammer (2012) is not possible as they only give confidence scores for parent-child attachments while our approach gives confusion scores for parent-child edge's dependency label. In a typed dependency framework both attachments and labels are significant and hence our approach is complementing Mejer and Crammer (2012).

5 Conclusion and Future Work

This paper presents our effort towards computing a confusion score that can estimate, upfront, the correctness of the dependency parsed tree. The confusion score, accredited with each edge of the output, is targeted to give an informed picture of the parsed tree quality. We supported our hypothesis by experimentally illustrating that the edges with a higher confusion score are the predominant parsing errors.

Not only parsed output, manual treebank validation too can benefit from such a score. An n-fold cross validation scheme can be adopted, in this case, to compute and assign confusion scores and detect annotation errors. Also, this score has scope in active learning where unannotated instances exhibiting high confusion can be prioritized for manual annotation.

References

Sabine Buchholz and Erwin Marsi. 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.

Chih-Chung Chang and Chih-Jen Lin. 2011. Libsvm: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):27.

Corinna Cortes and Vladimir Vapnik. 1995. Support-vector networks. *Machine learning*, 20(3):273–297.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. Liblinear: A library for large linear classification. *The Journal of Machine Learning Research*, 9:1871–1874.

Michel Galley and Christopher D Manning. 2009. Quadratic-time dependency parsing for machine translation. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 773–781. Association for Computational Linguistics.

Johan Hall, Jens Nilsson, Joakim Nivre, Gülşen Eryiit, Beáta Megyesi, Mattias Nilsson, and Markus Saers. 2007. Single malt or blended? a study in multilingual parser optimization. *Proceedings of the CoNLL Shared Task of EMNLP-CoNLL 2007*, pages 933–939.

Rebecca Hwa. 2004. Sample selection for statistical parsing. *Computational Linguistics*, 30(3):253–276.

Shunsuke Ihara. 1993. *Information theory for continuous system*, volume 2. World Scientific Publishing Company.

Christopher D Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to information retrieval*, volume 1. Cambridge University Press Cambridge.

Ryan T McDonald and Joakim Nivre. 2007. Characterizing the errors of data-driven dependency parsing models. In *EMNLP-CoNLL*, pages 122–131.

Ryan McDonald and Fernando Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *Proceedings of EACL*, volume 6, pages 81–88.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič. 2005. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 523–530. Association for Computational Linguistics.

Avihai Mejer and Koby Crammer. 2012. Are you sure?: confidence in prediction of dependency tree edges. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 573–576. Association for Computational Linguistics.

Jens Nilsson, Sebastian Riedel, and Deniz Yuret. 2007. The conll 2007 shared task on dependency parsing. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL*, pages 915–932. sn.

Joakim Nivre and Jens Nilsson. 2005. Pseudo-projective dependency parsing. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 99–106. Association for Computational Linguistics.

Joakim Nivre and Mario Scholz. 2004. Deterministic dependency parsing of english text. In *Proceedings of the 20th international conference on Computational Linguistics*, page 64. Association for Computational Linguistics.

Joakim Nivre, Johan Hall, Jens Nilsson, Gülşen Eryiit, and Svetoslav Marinov. 2006. Labeled pseudo-projective dependency parsing with support vector machines. In *Proceedings of the Tenth Conference on Computational Natural Language Learning*, pages 221–225. Association for Computational Linguistics.

Joakim Nivre, Johan Hall, Jens Nilsson, Atanas Chanev, Gülsen Eryigit, Sandra Kubler, Svetoslav Marinov, and Erwin Marsi. 2007. Maltparser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13(2):95.

John Platt et al. 1999. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers*, 10(3):61–74.

Martin Popel, David Mareček, Nathan Green, and Zdeněk Žabokrtskỳ. 2011. Influence of parser choice on dependency-based mt. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 433–439. Association for Computational Linguistics.

Dipti Misra Sharma, Prashanth Mannem, Joseph van-Genabith, Sobha Lalitha Devi, Radhika Mamidi, and Ranjani Parthasarathi, editors. 2012. *Proceedings of the Workshop on Machine Translation and Parsing in Indian Languages*. The COLING 2012 Organizing Committee, Mumbai, India, December.

Karan Singla, Aniruddha Tammewar, Naman Jain, and Sambhav Jain. 2012. Two-stage approach for hindi dependency parsing using maltparser. *Training*, 12041(268,093):22–27.

Min Tang, Xiaoqiang Luo, and Salim Roukos. 2002. Active learning for statistical natural language parsing. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 120–127. Association for Computational Linguistics.

Ting-Fan Wu, Chih-Jen Lin, and Ruby C Weng. 2004. Probability estimates for multi-class classification by pairwise coupling. *The Journal of Machine Learning Research*, 5:975–1005.

Hiroyasu Yamada and Yuji Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *Proceedings of 8th International Workshop on Parsing Technologies*, pages 195–206.

Yue Zhang and Stephen Clark. 2008. A tale of two parsers: investigating and combining graph-based and transition-based dependency parsing using beam-search. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 562–571. Association for Computational Linguistics.