Comparison of Parsers Dealing with Text Ambiguity in Natural Language Processing

Sareya Qayyum, Nimra Aziz, Waqas Anwar, Usama Ijaz Bajwa
Computer Science
COMSATS University Islamabad, Lahore Campus
{sareyaqayyum, nimraiftikhar1995}@gmail.com, {waqasanwar, usamabajwa}@cuilahore.edu.pk

Abstract

Parsing in Natural Language processing is a vast domain which serves as a pre-processing step for different NLP operations like information extraction, etc. Multiple parsing techniques have been presented until now. Some of them unable to resolve the ambiguity issue that arises in the text corpora. This paper performs a comparison of different models presented in two parsing strategies: Statistical parsing and Dependency parsing. The comparison has been made on very famous Penn Treebank corpus specifically involving its Wall Street Journal Portion.

1. Introduction

In Natural Language Processing (NLP), Parsing acts as an essential and key component to many problems. Parsing is the analysis of syntax or commonly called syntactic analysis in which we process the sentences following the rules of a formal grammar. Parsing involves uncovering the meaning, content and underlying structure that makes up a sentence. Every Language has a unique grammar thus making parsing process unique as well. Languages can be divided into two categories:

Segmented Languages: Those languages whose words are space-delimited e.g. English and Spanish language

Un-Segmented Languages: In un-segmented Languages word segmentation is required as a pre-processing step before further processing E.g. Chinese and Japanese Languages

A language is not just a 'bag of words' or else there would be no need for grammar. Grammatical rules apply to sentences where a sentence in a language is a group of strings that consists of two things: a subject and a predicate. A subject is defined as a Noun Phrase (NP) and predicate as a verb phrase (VP). E.g. In an English Sentence 'Sam went to school', 'Sam' is NP and 'went' is VP. Parsing has many applications in Natural Language Processing. For Example, Machine translation, text summarization and question answering are few areas of NLP in which immense work is being performed, etc. Parsing serves as an initial step for these problems. The result of parsing a sentence using a formal grammar is a tree structure. A sentence can have exactly one or many such tree structures. The grammar that is usually used is CFG (context-free grammar).

One of the major challenges in Parsing is dealing with ambiguities in the sentence. Ambiguity refers to sentences that are subjective, open to interpretation and can have multiple meanings. Three types of ambiguities are present in a sentence when parsing is performed namely Syntactic Ambiguity, Lexical Ambiguity, and Semantic Ambiguity.

Syntactic Ambiguity: Sentences can be parsed in multiple syntactical forms. E.g. 'I heard his cell phone ring in my office'. The Phrase 'in my office' can be parsed in a way that modifies the noun or vice versa modifies the verb.

Lexical Ambiguity: Sentences having Lexical ambiguity can have words with multiple assertions. E.g. 'book' is used as a noun when used in a sentence as 'He loves to read books'. On the other hand, it can also be used as a verb when used in a sentence as 'He books an appointment at the dentist'.

Semantic ambiguity: It is related to the interpretation of the sentence. E.g. 'I saw a man with the telescope'. It can be deduced as if I saw a man holding a telescope. Or I saw a man through a telescope. Parsing of sentences happens in multiple stages:

a) Dividing a sentence into tokens. These tokens are used as input to some other tasks like parsing.

b) Tagging each token with parts of speech.

Eight parts of speech are observed in the English language – verbs, nouns, pronouns, adverbs, adjectives, conjunctions, interjections and prepositions.

1.1. Types of parsing

Two main types of parsing will be discussed in this paper and a comparison of performances for both the types will be made. Following are the types:

i. Statistical Parsing
ii. Dependency Parsing

1.1.1. Statistical Parsing

In Natural Language processing statistical parsers are the type of parsers which associate grammar rules with probability. The Statistical parser is an algorithm that looks for a tree that maximizes the probability P (T|S). PCFG (Probabilistic

Context-Free Grammar) which is an extended form of CFG is used as an underlying grammar. The probability of a parse tree of a sentence can be computed by firstly calculating the probability of the productions used in the derivation of the tree and then taking the product of these probabilities. Following three main tasks are involved in statistical parsing:

1. Determine the likely parse trees for the sentence.
2. Assign probabilities to each derived parse
3. Select the most probable parse (highest probability)

Statistical parsing requires a corpus of hand-parsed text. For this purpose, we have Penn treebank (Marcus 1993). Penn Treebank is extensively used since it is the largest annotated dataset for English. In recent years Penn Treebank has been immensely used and considered as a standard for training and testing statistical parsers. Parseval measures are used to evaluate the Penn Treebank parsers. From many Parseval measures most commonly used ones are and labelled recall (LR) and labelled precision (LP). Sometimes Bracketed precision (BP) and bracketed recall (BR) are also used which are less strict measures then LP and LR.

1.1.2. Dependency Parsing

Identifying a sentence and allocating a syntactic arrangement to it is the major task of dependency parsing. In the dependency-based method, the head-dependent relation provides an estimate to the semantic relationship between arguments and their predicates.
The translation of a sentence to its dependency structure is done in two subtasks:

- Classify the structure for all head-dependent relationships.
- Classify these relations with their correct dependency relations.
-

A tree of dependency parsing is a coordinated diagram (a directed graph) which fulfills the stated following limitations:

- It consists of a single assigned root hub that does not have any approaching segments or arcs.
- With the exemption of the root hub, every vertex has precisely one approaching segment or arc.
- From every vertex in V, a unique way exits from the root hub.

In short, the stated requirements guarantee that every word has a distinct head, to which the dependency tree is linked, and a unique root hub by which one can pursue a unique guided path to each word of the sentence.

The idea of projectivity forces an extra restriction and is firmly identified with the setting free nature of human dialects. If there is a way from the head to each word that lies between the head and it's dependent, then an arc from a head to its dependent is considered as projective. A dependence tree is therefore said to be projective if every single one of the arcs is projective There are, in any case, numerous impeccably substantial developments especially in dialects with a generally adaptable word that leads to non-projective trees.

Presently the assignment of Syntactic parsing is very unpredictable because of the way that a given sentence can have numerous parse trees which we call as ambiguities. Consider a sentence "Book that flight." which can frame various parse trees dependent on its uncertain grammatical speech tags except if these ambiguities are settled. Picking a right parse from the numerous conceivable parses is called as syntactic disambiguation.
The two main methods of dependency parsing are:

a) Transition-based dependency parsing
b) MST (Maximum spanning tree) dependency parsing

Transition based parsers commonly have a linear or quadratic complexity. MST based parsers divides the dependency structure into small parts called 'factors'. The components of the principle MST parsing algorithm are edges that consists of the head, the edge name and the dependent (child). This algorithm has quadratic complexity. (Bernd Bohnet., 2010).
Treebanks have a critical job in the advancement and assessment of dependency parsers. Having human annotators legitimately create dependency structures for a given corpus. The most generally utilized syntactic structure is the parse tree which can be produced utilizing some parsing algorithms. These parse trees are valuable in different applications like sentence grammar checking, co-reference goals, question, and their answers, data extraction or all the more significantly it assumes a basic job in the semantic analysis stage. We can likewise utilize a deterministic procedure to decipher existing principal based treebanks into dependency trees using head rules. The significant English reliance treebanks have to a great extent been removed from existing assets, for example, the Wall Street Journal segments of the Penn Treebank (Marcus et al., 1993). The later OntoNotes venture (Hovy et al. 2006, Weischedel et al. 2011) expands this methodology going past customary news content to incorporate conversational phone speech, newsgroups, weblogs and talk programs in English, Arabic and Chinese.

2. Literature Review

Following is the previous work for statistical parsing and dependency parsing.

2.1. Statistical Parsing

Magerman [9] presented a statistical parser called SPATTER. It achieves the best accuracy by building a complete parse for every sentence in the corpus. SPATTER is based on a decision-tree learning technique. Using the PARSEVAL evaluation measure, SPATTER on the Penn Treebank Wall Street Journal corpus achieves 86% precision P (see equation 1) and recall R (see equation 2), and 1.3 crossing brackets CB (see equation 3) per sentence for sentences with a word length of 40 or less. For sentences having word length between 10 and 20, SPATTER achieves 91% P, 90% R, and 0.5 CB.

P = number of Correct Components / number of Components in parser output (1)

R = number of correct Components / number of Components in gold standard (2)

CB = number of Components in parser output that cross gold standard Components / number of Components in parser output (3)

In 1996 Collins [10] presents his first model for statistical parsing. Below equation 4 demonstrates a conditional model capable of parse selection, where for a given sentence S in the corpus, the probability of a parse tree T is calculated directly. Input to the model is a Part of Speech (POS) tagged sentence which produces a tree as an output. For a given sentence S in the corpus and its tree T, the conditional model for Collins represents the probability in the following manner:
The most likely parse under the model is then:

$$T(\text{best}) = \text{argmax}\ T\ (P\ (T\ |S)) \quad (4)$$

Collins (1996) model showed an improvement in Precision and Recall when compared with Magerman's (1995) results.

Collins presented a new model in 1997 [11] which improved on the previous results of Collins conditional model (1996). This approach is based on a generative model. The Collins (1997) accounts for word-word dependencies when generating a parse tree. This model focuses on the modeling of the parses and deals with the flat trees of the Penn Treebank corpus. This generative version of Collins parser corpus shows an improvement of 2.3% on the conditional model of Collins (1996). It achieves 88.1% P and 87.5% R on Wall Street Journal.

In [12] Collins (1999) few problems were observed with the generative model for Collins (1997). It was noticed that Collins model is no longer a model for predicting maximum likelihood because of how the dependency probabilities were estimated. Another deficiency is its way of considering all dependency relations as independent. Due to these reasons, Collins presented another modification to his previous model.

Charniak [13] (2000, 1999) after presenting his first model for statistical parsing in 1997 Charniak presented another statistical Treebank parser. It outperformed the Collins generative model presented in 1997. Charniak's main advantage to Collin's is generating candidate parses using a simple probabilistic chart parser. Charniak's model showed an improvement of 0.45% in labelled recall (LR) and labelled precision (LP). The model achieved average Precision and an average recall of 91.1% on sentences with length less than 40. For sentences with length less than 100, the model achieved 89.5% average precision and recall on Penn Treebank corpus. Over the previously best results, Charniak's model achieves an error reduction of 13% for single parser on this test set.

Henderson and Brill [14] presented an approach where they combined the results/prediction of three current existing parsers. They combined Charniak's 1997 model with Collins 1997 and Ratnaparkhi 1998 model to better understand the capabilities of parsers and to check if they yield better results. This combination of three parsers gave the best results with Labelled precision of 92.1% and Labelled Recall of 89.2% on the development set while achieving LP of 92.4% and LR of 90.1% on the test set of Penn Treebank. These are best-known results up till now.

Parser presented by Bod [15] claims to give a better performance in terms of Parseval measures. It improved on the Charniak's result by achieving 89.7% LP and LR on sentences with 100 words. For sentences with 40 words or less Bod's model achieves 90.8% LP and 90.6% LR. Although it is debatable whether an increase of 0.2% in LP and 0.1% in LR is considered an improvement. Bods' model takes arbitrary structural and lexical dependencies into consideration when computing probabilities of a parse tree as it is based on Data-Oriented Parsing (DOP).

Collins in 2000 and Collins and Duffy in 2002 [16] presented two approaches in which they improved the performance for Collins 2000 model by re-ranking the parses using a different model on the outcome of Collins' 1999 model. LP improved from 88.1% to 88.3% while LR improved from 88.3% to 89.6% on Collins 1999 model. For Collins 2000 model using a variant for boosting LP improved to 88.3% and LR to 89.6%. For Collins and Duffy 2002 model by using a DOP like approach using a voted Perceptron LP improved to 88.6% and LR to 88.9 %.

2.2. Dependency Parsing

A huge interest has been seen in Dependency Parsing lately for applications such as relation extraction, machine translation, synonym generation, and lexical resource augmentation. The main reason for using dependency structures is because they are highly effective to study and parse while still training much of the predicate-argument information needed in a lot of applications.

Most of these parsing models have concentrated on trees that are projective, including the effort of Eisner (1996), Yamada and Matsumoto (2003), Collins et al. (1999), Nivre and Scholz (2004), and McDonald et al. (2005).
A parsing model presented by Nivre and Nilsson in 2005 allows to include edges that are non-projective, into trees using learned edge transformations in the memory-based parser. The method varies in examining efficiently the full span of non-projective trees. The main focus was that the dependency parsing can serve as the main search point for an MST in a directed graph. This specifies the regular projective models of parsing that are based on the Eisner algorithm (Eisner, 1996) to have a better efficient of O (n 2). By using the spanning-tree illustration, to cover non-projective dependencies we extend the work of McDonald et al. (2005) on online large-margin discriminative training methods (McDonald, Pereira, and Ribarov; 2005)

Nowadays there has been an increase in the usage of dependency representations through many tasks of natural language processing (NLP). Stanford dependency is extensively used in both NLP and biomedical text mining.

Stanford Dependency was originally extracted from constituent parses but the production of parse trees from the raw text was quite time. The approaches hence designed specifically for dependency parsing such as Covington, minimum spanning tree (MST), Eisner and Nivre should perform faster, assuming that they have low time complexity. The different approaches are compared in terms of their collective accuracy and speed and characteristic errors are reported. The parsing models are trained using the training set extracted from the Penn Treebank that consists of sections 2 through 21, different parsers of dependency were compared such as MaltParser package v1.3 selected models, the MSTParser 0.4.3b, and the rule Based RelEx parser 1.2.0. We used F1-score other than accuracy because the typical Stanford dependency representation parsers can generate a variety of different dependencies for every sentence. The fastest parsers were the Malt package, Nivre, and Covington. Nivre Eager and MSTParser (Eisner) and they achieved better F1scores when the interaction between model and features was not used. If parsing a huge amount of data, and speed is important, the experiments suggest that the top choice is to use parsers included in the Malt package. (Cer, D. M., De Marneffe, (2010, May)).

3. Performance Evaluation

Many different parsers have been presented in the above section having unique abilities and different performances each trying to perform well than the previous. The main question is what type of parsers should be used in which context. The decision to apply these models depends on the type of the problem under study. To facilitate the decision making a performance overview of all the parsers is given below.

3.1. Statistical Parsing

Table I shows the performance of multiple Statistical Parsers over the Parseval Measures.

• In [9] it was shown that previous syntactic natural language parsers used were not capable of handling ambiguous large-vocabulary text. They had poor performances on standard datasets like Wall Street Journal of Penn Treebank which led to the new approach presented by Magerman called SPATTER (Statistical Pattern Recognizer). It is based on decision-tree learning technique and has accuracy way better than any parser published up till 1995. SPATTER requires very less linguistic knowledge and is compared against state of the art grammar-based parsers. Decision trees provide a ranking system by assigning a probability distribution to the possible choices, which not only specifies the order of preference but also gives a measure of the relative likelihood that each choice is the one which should be selected. The limitation of the searching strategy of SPATTER is its possible consumption of available memory before completing the search. But conveniently this memory exhaustion occurs on sentences which SPATTER most likely will get wrong anyway. So little or no performance loss is observed due to this search errors.

• The study in [10] reveals that Collins and Magerman both use lexicalized PCFG which is associating a head word to every non-terminal present in the parse tree. Collins parser performs almost equally as SPATTER when it is trained and tested on Wall Street Journal portion of Penn Treebank. The advantage of Collins 1996 model over SPATTER is the simplicity of its architecture and working. Still, many improvements can be made by using a more sophisticated probability estimation techniques like deleted interpolation or estimation on relaxing the distance measure for smoothing could be used. Another limitation of Collins 1996 model is that it does not account for valency when calculating the parse.

• Collins [11] attempt to address the flaws of the model presented in 1996 by putting forth 3 models. It shows that sub-categorization and wh-movement can be given a probabilistic treatment thus resulting in the statistical interpretation of the concepts causing an increase in performance by adding useful information to the parser's output. The average improvement of Collins 97 over the previous model is 2.3%. Model 1 presented has clear advantages when handling unary rules and distance measures. Model 2 and 3 can apply condition on any structure that has been previously generated while Collins 96 lack in this treatment.

• [12] Is an update of previous works of Collins. It addresses the limitation of Collins 1996 and 1997 related to punctuation as surface features of the sentence. Previous models failed to generate punctuation and are considered a deficiency of the model. Collins 2000 uses a technique that is based on boosting algorithms for machine learning for re-ranking the best outputs using additional features.

• The major invention of Charniak's [13] 2000 model is the use of maximum entropy inspired model which results in an increase of 2% in performance due to its strategy of smoothing and to combine multiple conditioning events for testing. Maximum entropy inspired approached has certain advantages over the probabilistic model and has recommended itself for use due to its novel approach of smoothing. Most important progress accomplished by using Charniak's model over conventional deleted interpolation is the flexibility achieved due to simpler maximum-inspired-model which let us experiment with different conditioning events and to move up to Markov grammar without significant programming. This model uses Markov processes to generate rules. The additional features incorporated boost the performance. The main goal for Charniak's parser is to generate model flexible enough to allow changes for parsing to more semantic levels.

• To solve some of the fundamental problems of Natural Language processing like parsing some authors including Henderson and Brill [14] may adopt a unique approach to combine the previous parsers to obtain better results. Collins along, with Charniak and Ratnaparkhi model, are experimented to explore different parser combination. With poor parser being introduced during the experiments, Techniques like parser switching and parser hybridization still gave better results. For more powerful parser combinations the results can be improved further.

• Bod 2001 [15] presents results that are comparable to the results of previous models like Charniak and Collins 2000. The main goal of the Bod model is to achieve maximal parse accuracy by applying constraints of several words in a fragment and to the depth of lexicalized fragments. Many previous models applied constituent lexicalization on Wall Street portion of Penn Treebank while Bods 2001 DOP based model uses frontier lexicalized approach. The results obtained from Bods models claim that using frontier lexicalization yields better results and is a better alternative to constituent lexicalization. Another difference of Bod with other models is its use of treebank grammar as an underlying grammar of its DOP model. Another future area could be the application of Markov grammar on the DOP model which will further improve the results.
• The main advantage of this model presented by Collin and Duffy [16] is the application of the perceptron algorithm on exponentially big representations of parse trees. It is computationally efficient and leads to a polynomial-time 2 algorithm for training and testing phases of the perceptron. It can stretch to more complex domains. Due to its different parameter estimation when compared to Bod and other models the computation is manageable.

TABLE I: Performance of all statistical parsers on penn treebank corpus

Parsers	Evaluation Measures		
	LP	***LR***	***CB***
Magerman 1995 [9]	86%	86%	1..3
Collins 1996 [10]	86.3%	85.8%	1.14
Collins 1997 [11]	88.6%	88.1%	0.96
Henderson and Brill [14]	92.4%	92.1%	-
Collin 2000 [12]	90.4%	90.1%	0.73
Charniak 2000 [13]	90.1%	90.1%	0.74
Bod 2000 [15]	90.8%	90.6%	-
Colins and Duffy 2001 [16]	88.6%	88.9%	-

3.2. Dependency Parsing

3.2.1. Deterministic Dependency Parsing

Collins and Charniak are one of the best accessible parsers prepared on the Penn Treebank, utilize statistical models for disambiguation that utilize dependency relations. The Yamada and Matsumoto method of deterministic dependency parser and that of Collins and Charniak, when prepared On the Penn Treebank, gives a nearly equal accuracy. The parser depicted in this paper is like that of Yamada and Matsumoto in that it utilizes an algorithm of deterministic parsing in blend with a classifier actuated from a treebank. Be that as it may, there are likewise significant differences between the two methodologies. Most importantly, while Yamada and Matsumoto utilizes a severe algorithm of bottom-up(basically shift-reduce parsing), the present parser utilizes Nivre's algorithm, which uses bottom-up and top-down approaches together to increase the accuracy. The experiment was carried out on two sets whose result is shown in table II and IIA.

- Set G which contained grammatical roles from Penn
- Set B contained the function tags for grammatical roles with normal bracket labels (S, NP, VP, etc.).

And the evaluation metrics used are:

- Unlabeled attachment score is the measure of words that are root and are correctly identified as head.
- Labelled attachment score is the measure of words that are root and are correctly identified as head and their dependency type.
- Dependency accuracy is the measure of words that are non-root and are correctly identified as heads.
- Root accuracy is the measure of root words correctly identified as roots.
- Complete match is the measure of sentences whose unlabeled dependency structure is correctly identified.

TABLE II: PERFORMANCE OF DETERMINISTIC DEPENDENCY PARSERS ON PENN TREEBANK CORPUS (6. NIVRE AND M.SCHOLZ 2004)

Parsing Models	Evaluation metrics		
	Dependency Accuracy	***Root Accuracy***	***Complete Match***
Charniak	92.10%	95.20%	45.20%
Collins	91.50%	95.30%	43.30%
Yamada and Matsumoto	90.30%	91.60%	38.40%
Nivre and Scholz	87.30%	84.30%	30.40%

TABLE IIA: PERFORMANCE OF DETERMINISTIC DEPENDENCY PARSERS ON PENN TREEBANK CORPUS (6. NIVRE AND M.SCHOLZ 2004)

Evaluation metrics	Data sets		
	Grammatical Roles from Penn II (Experiment # 1)	***Function Tags for Grammatical Roles (Experiment #2)***	***Combination of both Experiments***
Unlabeled Attachment Score	85.8%	87.1%	-
Labelled Attachment Score	84.6%	84.4%	86.0%

3.2.2. Constituent-to-Dependency Parsing

PENN2MALT disposes of the deep information in the dependency tree. In the new strategy, the topicalized phrases and words are connected to their respective semantic head. Other than this the new approach used a richer collection of arced labels than that used in PENN2MALT. MALTPARSER depends on a parsing system that constructs a parse tree gradually while continuing through the sentence one token at any given moment. By utilizing this type of system, a rich history-based list of capabilities for the SVM classifier is made, that can be used for choosing activities. MSTPARSER predicts a parse tree by expanding a function of scoring over the space of all parse trees. The scoring function is a weighted sum of single connections or links. Table III describes MALT Parsers and MST Parser for different parsing sets.

TABLE III: PERFORMANCE OF MALT AND MST DEPENDENCY PARSERS ON PENN TREEBANK CORPUS (17.JOHANSSON,R., & NUGUES,P.(2007))

Parsing sets	Parsing models			
	MALT PARSER	***MALT PARSER***	***MST PARSER***	***MST PARSER***
	LABELED	***UN LABELED***	***LABELED***	***UNLABELED***
PENN2 MALT	90.30%	91.36%	92.04%	93.06%
NEW CONV E-RSION	87.63%	90.54%	86.92%	91.64%

3.2.3. MIRA (18. McDonald, R., Crammer, K., & Pereira, F. (2005))

It is an online learning algorithms which intuitive and easy to comprehend and implement. To form dependency structures the extraction rules of Yamada and Matsumoto were used. For the evaluation and development of sets, the tagging system of Ratnaparkhi was used and POS tags were assumed as the input for the system.

For large margin multi-class classification, Crammer and Singer established an approach (equation #5) which was then extended to structured classification by Taskar.

EQUATION # 5 (18. MCDONALD, R., CRAMMER, K., & PEREIRA, F. (2005))

$$\begin{aligned} &\min \|\mathbf{w}\| \\ &\text{s.t.} \quad s(\boldsymbol{x}, \boldsymbol{y}) - s(\boldsymbol{x}, \boldsymbol{y}') \geq L(\boldsymbol{y}, \boldsymbol{y}') \\ &\forall (\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{T},\ \boldsymbol{y}' \in \mathrm{dt}(\boldsymbol{x}) \end{aligned}$$

The above-mentioned equation #1 optimization is directly mapped into the online framework by the margin infused relaxed algorithm (MIRA). On every attempt, while applying the change to the parameter vector MIRA tries to maintain the standard and keep it as small as possible, to classify an instance correctly the margin should be at least equal to the loss of classifying incorrectly. This can be done by substituting the following changes in the original algorithm (equation #5) present online and form another (equation #6).

EQUATION # 6 (18. MCDONALD, R., CRAMMER, K., & PEREIRA, F. (2005))

$$\begin{aligned} &\min \left\|\mathbf{w}^{(i+1)} - \mathbf{w}^{(i)}\right\| \\ &\text{s.t. } s(\boldsymbol{x}_t, \boldsymbol{y}_t) - s(\boldsymbol{x}_t, \boldsymbol{y}') \geq L(\boldsymbol{y}_t, \boldsymbol{y}') \\ &\forall \boldsymbol{y}' \in \mathrm{dt}(\boldsymbol{x}_t) \end{aligned}$$

To apply dependency parsing using MIRA, we can just consider the parsing as a multi-class classification problem in which, every dependency tree is considered as one of the classes of the sentence. Nevertheless, this clarification fails in reality as a normal sentence has a lot of possible dependency trees thus making it exponentially complex. To overcome this issue another equation #7 was created.

EQUATION # 7 (18. MCDONALD, R., CRAMMER, K., & PEREIRA, F. (2005))

$$\begin{aligned} &\min \left\|\mathbf{w}^{(i+1)} - \mathbf{w}^{(i)}\right\| \\ &\text{s.t.} \quad s(\boldsymbol{x}_t, \boldsymbol{y}_t) - s(\boldsymbol{x}_t, \boldsymbol{y}') \geq L(\boldsymbol{y}_t, \boldsymbol{y}') \\ &\forall \boldsymbol{y}' \in \mathrm{best}_k(\boldsymbol{x}_t; \mathbf{w}^{(i)}) \end{aligned}$$

TABLE IV: PERFORMANCE OF MIRA DEPENDENCY PARSERS ON PENN TREEBANK CORPUS

Parsing Models	Evaluation measures		
	Accuracy	***Root***	***Complete***
Y&M	90.30%	91.60%	38.40%
N&S	87.30%	84.30%	30.40%
AVERAGE PERCEPTION	90.60%	94.0%	36.50%
MIRA	90.90%	94.20%	37.50%

Accuracy: number of words whose parents are correctly identified.

Root: number of trees in which the root is identified correctly.

Complete: number of sentences whose dependency was correctly identified.

4. Conclusion

Comparison between parsers leads us to examine the similarities and differences between multiple models and the scenarios in which they tend to perform better. The famous CKY parsing algorithm can represent the ambiguities that occur while parsing efficiently but it is not able to resolve them. So statistical and dependency models are designed to overcome the Limitations of the previous ones thus showing an increase in the overall measures of evaluation.

5. Acknowledgment

We would like to give our deepest appreciation to all the people who helped us complete this paper. A special gratitude to our instructor and co-author of this paper [Dr. WAQAS ANWAR], whose support helped us in achieving the goals of writing this paper.

6. References

[1] Marcus, B. Santorini, and M. Marcinkiewicz. 1993. Building a large annotated corpus of English: the Penn Treebank. Computational Linguistics, 19(2):313–330.

[2] Snow, D. Jurafsky, and A. Y. Ng. 2004. Learning syntactic patterns for automatic hypernym discovery. In NIPS 2004.

[3] Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In Proc. COLING.

[4] Yamada and Y. Matsumoto. 2003. Statistical dependency analysis with support vector machines. In Proc. IWPT.

[5] Nivre and M. Scholz. 2004. Deterministic dependency parsing of English text. In Proc. COLING

[6] McDonald, Pereira, Ribarov. 2005. Non-projective Dependency Parsing using Spanning Tree Algorithms.

[7] McDonald, R., & Pereira, F. (2006). Online learning of approximate dependency parsing algorithms. In *11th Conference of the European Chapter of the Association for Computational Linguistics*.

[8] Magerman, D. M. (1995, June). Statistical decision-tree models for parsing. In *Proceedings of the 33rd annual meeting on Association for Computational Linguistics* (pp. 276-283). Association for Computational Linguistics.

[9] Collins, M. J. (1996, June). A new statistical parser based on bigram lexical dependencies. In *Proceedings of the 34th annual meeting on Association for Computational Linguistics* (pp. 184-191). Association for Computational Linguistics

[10] Collins, M. (1997). Three generative, lexicalized models for statistical parsing. ArXiv preprint cmp-lg/9706022.

[11] Collins, M. (2003). Head-driven statistical models for natural language parsing. Computational linguistics, 29(4), 589-637

[12] Charniak, E. (2000, April). A maximum-entropy-inspired parser. In Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference (pp. 132-139). Association for Computational Linguistics.

[13] Henderson, J. C., & Brill, E. (2000). Exploiting diversity in natural language processing: Combining parsers. ArXiv preprint cs/0006003

[14] Bod, R. (2001, July). What is the minimal set of fragments that achieves maximal parse accuracy? In Proceedings of the 39th Annual Meeting on Association for Computational Linguistics (pp. 66-73). Association for Computational Linguistics.

[15] Collins, M., & Duffy, N. (2002, July). New ranking algorithms for parsing and tagging: Kernels over discrete structures, and the voted perceptron. In Proceedings of the 40th annual meeting on association for computational linguistics (pp. 263-270). Association for Computational Linguistics

[16] McDonald, R., Crammer, K., & Pereira, F. (2005). Online large-margin training of dependency parsers. In Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05) (pp. 91-98).

[17] Johansson, R., & Nugues, P. (2007). Extended constituent-to-dependency conversion for English. In Proceedings of the 16th Nordic Conference of Computational Linguistics (NODALIDA 2007) (pp. 105-112)