11-712: NLP Lab Report
TamilDep : A dependency parser for Tamil.

Pradeep Prabakar Ravindran

April 26, 2013 (Final Due date)

Abstract

This report documents various phases in building a dependency parser for Tamil. First, a completely new dependency corpora is built totally consisting of around 6350 tokens. Two separate corpora (Corpus A and B) each containing 1000 tokens are then set aside as test data for evaluating the performance of various parsing approaches. The raw data that is used for annotation comes from the English Tamil parallel corpora (Ramasamy et al. (2012)) that consists of sentences from news articles, cinema and bible passages. Next, various approaches for parsing including supervised and unsupervised approaches are discussed. The final model that uses morphological features and clusters of each lexical item gives an attachment accuracy of about 69.7 on Corpus A and 67.9 on Corpus B.

TamilDep is a tool that performs dependency parsing of the Tamil Language. This tool would soon be open-sourced and become publicly available at `https://github.com/pradeep-gnr/TamilDep`. Along with the code for a dependency parser, a corpora of tamil sentences annotated with their dependency parse trees is also expected to be released. Since there are very few resources publicly available for Tamil, I hope that this software would prove useful to researchers working on Tamil natural language processing.

1 Basic Information about Tamil

Tamil is a Dravidian language that is widely spoken in the Indian state of Tamil Nadu and the north eastern part of Sri Lanka. Reports estimate that there are about 70 million native Tamil Speakers all over the world (Wikipedia (2013)). In terms of history, Tamil is also one of the oldest languages in the world with works of Tamil literature dating back to almost 2000 years ago (Wikipedia (2013)).

Grammatically, Tamil has a lot of characteristics that make it pretty different from the Germanic languages. For example, Tamil almost always follows a subject-object-verb oriented form (Ramasamy and Žabokrtský (2011)). For instance, consider the following sentence.

`Avan`	`angu`	`selgiran`
`(He)`	`(there)`	`(going)`

The sentence approximately translates to 'He is going there'. Sentences where inflected verb forms occurring at the end of the sentence are relatively common in Tamil. Also, similar to languages like Czech, Tamil is relatively a free word order language. For example, the sentence that was discussed above could be expressed in multiple forms such as

angu	avan	selgiran
(There)	(he)	(going)

selgiran	avan	angu
(going)	(he)	(there)

All of these sentences are perfectly valid in terms of grammatical structure. Another key characteristic of Tamil sentences is that words could be agglutinative and sometimes it is very difficult to even define what a word is. There are a number of ways in which suffixes could be added to the noun and verb stems to produce highly inflected forms.

For instance, 'has gotten over' which is a phrase in English could be expressed as a single word in Tamil. Also, there could be multiple such words that would approximately translate to the sample phrase. Some such examples are discussed below.

```
1) Nadanthathu
2) Nadanthathuvittathu
3) Nadanthayittru
```

All these words express the phrase 'has gotten over' and they are all highly inflected forms of the root word 'Nada' which roughly translates to 'happening' or 'proceeding'. Since these type of verb phrases are relatively common in Tamil, identifying the 'root' of the head words of such phrases would be a challenge from a dependency parsing perspective.

2 Past Work on the Syntax and NLP tools for Tamil

One of the most infleuntial books on Tamil grammar written in English was by Thomas Lehmann : A grammar of modern Tamil (Lehmann et al. (1989)). The book gives a very detailed analysis of several of the morphological and syntactic phenomenon that characterize written Tamil. While languages like English do not differ significantly from their spoken and written form, there are significant differences among the spoken and written forms in Tamil. Tamil has the property that sentences could be semantically valid even when certain words are omitted and spoken Tamil usually consists of a lot of missing words. Written Tamil however is more morphologically richer than its spoken version. Lehmann also notes that Tamil is relatively a head final language and that the phrasal head appears in several inflected forms at the end of the phrase (Lehmann et al. (1989)). Also, another key characteristic of Tamil is that the verbs agree with gender, tense etc. One of the key differences in tamil syntax when compared to other languages is that the language exhibits a restricted free word order. Hence there are a multitude of semantically and syntactically valid sentences that could be formed using the same set of words. From a parsing perspective - this property poses a lot of challenges because unlike English, writing context free grammar rules to characterize Tamil Syntax is a non-trivial problem. (Srinivasan CJ (2007)) discuss the challenges they face because of this issue while building a spoken dialog system for Tamil and turn to dependency parsing to aid semantic annotation of sentences.

Unlike other Indian languages like Hindi, Telugu and Bengali - Tamil does not have a lot of annotated corpora for NLP tasks. The Tamil dependency treebank ((Ramasamy and Žabokrtskỳ (2011))) which was released recently is the only publicly available dependency corpora that I could find online. The authors have also released an annotation manual that describes the rules that were used in labeling dependency edges among words. The entire corpus was annotated in the

Prague dependency treebank (PDT) (Böhmová et al. (2003)) format and a detailed description of this corpora and the annotation methodology that was used will be discussed in part 3. Although this corpus is quite small (about 3000 words), the annotations are quite diverse and cover various types of sentences in Tamil. The same group also used this corpus to build a rule based and corpus based dependency parser and report a 75 % accuracy when sentences are labeled with POS tags.

There was also an earlier paper (Dhanalakshmi and Rajendran (2010)) that used morphological analysis along with various custom heuristics for syntactic parsing of Tamil to identify phrasal consituents. Besides the work done by Ramaswamy et al (Ramasamy and Žabokrtskỳ (2011)), I could find only one related paper that describes a set of Natural language tools that were created specifically for Tamil. They have also released an open source Morphological analyzer for Tamil but unfortunately there was no dependency tree annotated corpora or software that was available for download. To the best of my knowledge, TamilDep would be the first open source dependency parser written specifically for Tamil.

3 Available Resources for Tamil

First of all to build a good annotated corpus, we need good quality sentences in the target language. The source text must be of high quality both in grammar and content. The content must also be diverse in the number of topics covered to make the parser more robust while handling different types of inputs. Tamil has a separate Wikipedia portal where there are quite a significant of documents that have been written exclusively in Tamil. There are also several other corpora that have been publicly released for various Tamil NLP tasks such as English-Tamil machine translation (Ramasamy et al. (2012)), Tamil Wordnet etc (Rajendran et al.).

The English-Tamil parallel corpora released by (Ramasamy and Žabokrtskỳ (2011)) consists of a total of around 169871 sentences that have been crawled from various news articles discussing multiple topics. From this corpus, I have extracted random sentences and constructed corpora A and B. Corpora A consists of about 70 sentences with 1126 tokens and corpora B contains 70 sentences with 1185 tokens. Additionally I have also created corpora C, which could be used as training data if I decide to pursue a supervised approach to dependency parsing. Corpora C consists of about 210 sentences and 3500 tokens. The data has been uploaded in the corpora folder in the root repository.

To aid the dependency annotation process, I decided to extend on the work done by (Ramasamy and Žabokrtskỳ (2011)) In their earlier paper on dependency parsing experiments for Tamil, (Ramasamy and Žabokrtskỳ (2011)) describe their experiences creating a detailed annotation corpora for Tamil. The annotations are in the Prague treebank format that has a three level layer based annotation at the syntactic and lexical level. They have also publicly released their annotation manual and rules to guide dependency annotation for Tamil. This manual is very comprehensive and documents several phenomena that is unique to Tamil and issues that a person needs to consider while performing the annotation. Because of the highly inflected nature of the language, morphological analysis must also be performed prior to syntactic annotation. The manual also discusses strategies for morphological analysis and outlines the rules in detail so that it could be easily extended into a morphological analyzer. I decided to use the custom tag set that was used by (Ramasamy et al. (2011)) in the hope that future annotated corpora produced for Tamil would conform to a particular standard and the research community can evaluate their methodologies on different corpora. Also, this would encourage the development of supervised models which tend to work well when more data is available.

4 Survey of Phenomena in Tamil

In this section, I shall describe some of the interesting phenomena that Tamil part of speech tags exhibit. Lehmann (Lehmann et al. (1989)) notes that Tamil has about eight part of speech tags (Nouns, Verbs, Postpositions, Adjectives, Adverbs, Quantifiers, Determiners, and Conjunctions). Each of these part of POS tags exhibit some interesting behavior that help characterize Tamil language.

4.1 Nouns

Lehmann states that nouns in Tamil are those words that can take case suffixes and also the suffixes (aay, aaka). The suffixes (aay, aaka) are adverbial suffixes that could be applied to several nouns. Lehmann also notes that not all nouns could take these suffixes and these nouns are called as defective nouns. Nouns in Tamil could also be inflected in a number of ways according to case and number. An inflected noun is usually of the form noun + (number) + case. The most common forms of suffixes that could inflect Tamil nouns are given below.

1. plural suffix
2. oblique suffix
3. euphonic suffix
4. case suffix

Noun stems in Tamil are the stems of nouns as they would be listed in a dictionary. Tamil noun stems could both be simple and complex. Complex nouns are however formed by combining a root and a derivational suffix. For example, paal (milk) is a simple noun having no suffixes while pati + ppu (study) is an example of a complex noun that has a verbal root (padi) and a suffix attached to it. Both these types are relatively common in Tamil. Tamil also exhibits a class of noun stem called as oblique stem that has no meaning when it exists by itself but can combine with case suffixes and post positions to produce various noun forms.

4.2 Pronouns

Tamil has the interesting phenomena that verbs with gender and case markings can stand out as separate sentences even when there is no explicit subject. As I discussed in the earlier sections, verb phrases in Tamil inflected with pronoun markers are relatively frequent in Tamil. For example, the verb 'walk' can be inflected as pronouns

1. Nadanthan - (He walked)
2. Nadandhal - (She walked)
3. Nadandhadhu - (It walked)

4.3 Postpositions

Postpositions are very important in Tamil and occur very frequently after nouns resulting in complex inflected phrases. Lehmann notes that postpositions could be inflected or un-inflected nouns or they could also be non-finite verb forms. Lehmann also discusses 8 different scenarios in which postpositions occur and I have described some of them below.

1. Occurring after nouns (Nominative)- (kaadu + varai - (till the forest)))
2. After nouns (accusative case) - (padippai + patri - (About your education))
3. Occurring after nouns (dative case) - (Idharku + paatilaga (Instead of this))

4.4 Adjectives

Lehmann notes that adjectives in Tamil usually are of two distinct types: simple and derived. Some examples of simple and derived adjectives are given below.

1. ketta paiyyan (bad boy)
2. vayathana aasiriyar (old teacher) - Here the adjective 'vayathana' consists of a noun stem 'vayathu' (age) and the suffix 'aana'.

   Derived adjectives are usually obtained by adding suffixes like 'aana' etc to nouns.

5 Initial Design

To perform the annotation, I have tried to follow the rules that were discussed in the annotation manual by Ramaswamy et al (Ramasamy et al. (2011)). The manual consists of several example sentences that cover a wide range of phenomena which was useful for me in making some decisions. One interesting phenomena, was that several Tamil sentences contained english words in between. To make my parser more robust, I have also considered such sentences. In most of the examples that I annotated, the head word mostly occurred at the end of the sentence. The annotated corpora A anb B have been uploaded in the repository. Each of them contain around 1000 tokens. Currently, I have just added the head information for each term in CONLL format. I have also included the text of the string in addition to the annotations. But as of now, I have not extracted any features such as POS tags and morphological information to the output. I could find only one open source POS tagger for Tamil (Dhanalakshmi and Rajendran (2010)) and I was unable to get the software to work. But by the next deadline, I think that I would be able to use the POS tag features as input for my model. Also, I am aware that features based on morphological analysis would be immensely useful for highly inflected languages such as Tamil, but again I was unable to get the morphological analyzer that was built by the same group (Dhanalakshmi and Rajendran (2010)) to work. I hope to resolve some dependency issues in the software to extract POS tags and morphological features before building my model. My idea is to build a supervised dependency parser by annotating some additional training data. I feel that supervised approaches would work well because I think more training data would help capture the wider range of morphological phenomena that occurs in Tamil words. I am currently annotating more data for the supervised model and I plan to use MaltParser for training the model. My eventual goal would be to build a stable data driven supervised dependency parser guided by a rich feature set such as POS tags and morphological annotations.

5.1 Annotation guidelines

In this section, I shall describe some of the annotation decisions that I followed. Some of these decisions were based on the syntactic structure of Tamil while other decisions were made to accommodate for tokenizer specific errors.

1. When a compound noun such as a named entity is encountered, the last word in that named entity would be the head of the preceding words. For example, for the named entity, 'Musharraf Khan', 'Musharaff' would be attached to 'Khan'. Similarly, I tend to follow this convention for locations and other named entities.

2. Punctuations such as comma and full-stop are attached to the word that precedes it.

3. When compound clauses occur in quotes, the main head in the quoted expression will be attached to the main head in the clause outside the quote. For example, in the sentence, '"Angu edharku selgirirgal", endru sonnan.', 'selgirargal' is the head of the quoted clause and 'sonnan' is the head of the sentence. Hence according to my rule, 'selgirargal' will be attached to 'sonnan'.

4. Quotations which are symbols attach to the word preceding it.

5. Sometimes, the tokenizer splits compound words into simple tokens based on punctuations or other that occur in the middle of the word. For example, the word 1950'yilIrundhu. The tokenizer that I use splits this word into (1950,',yilIrundhu). yilIrundhu acts like a postposition here and has no semantic bearing on its own. In such cases, I followed a convention that states that the postposition would be the head to its original attachments. For example, 'yilIrundhu' would be the head and 1950 would be attached to it.

6. When conjunctions connect two entities, the head of the conjunction will the second entity that it connects. For example, in the phrase "John mattrum Jack", the word 'mattrum' is a conjunction and in this case its head would be 'Jack'.

7. When consecutive adjectives occur, both the adjectives will connect to the corresponding noun they describe. For example, in the phrase 'Azhagana, thairiyamana pen (beautiful bold girl)' - The head of both "azhagana" and "thairiyamana" will be "pen".

8. RB POS tag words like (Maaraga, atharkaga) usually connect to head verbs in the corresponding clause.

6 System Analysis on Corpus A

For the baseline system, I have used a supervised learning approach where I had annotated additional 2000 tokens as training data. I found a Tamil chunker tool released by the Language Technologies Research centre at IIIT Hyderabad that performs various pre-processing tasks for Tamil (`http://ltrc.iiit.ac.in/showfile.php?filename=downloads/shallow_parser.php`). For example, it can automatically convert various English letters occurring in Tamil to an equivalent Tamil script. Also, it has an inbuilt POS tagger and a morphological analyzer in addition to a tokenizer. For the baseline, I just used the part of speech tags as features in the MALT parser. This baseline would give me a vague idea of how good/bad my approach is. Although, I have just used limited training data for the baseline - I am planning to annotate more training data and use roughly about

5000 tokens for training.

I used the standard settings prescribed in the Malt Parser and the unlabeled attachment score accuracy that I got was **43%**. Also, I tried another experiment where I tested the model built on the same training set. The accuracy was only 52% even in this case. This suggests that there is very few training data for the model to actually learn good rules from. The performance of the model is relatively poor and it was something that I had anticipated. The training data that I used was relatively small for the testing data that I used and that I think is one of the major reasons for this poor performance.

7 Lessons Learned and Revised Design

I think that one of the main drawbacks of my previous model was the lack of enough tokens for training. Also, since Tamil is a morphologically rich language, different morphological variations of words in the sentences can induce different parses even when two sentences have almost identically the same POS tags. So, it was essential that I annotate more training data that could cover sentences of different morphological variations. Apart from that I also tried to figure out other systematic errors by analyzing the predictions on the test set. The majority of errors that I could find were in long clauses that contained several nouns. In Tamil, nouns occur continuously in several sentences. However, these nouns are not simple nouns and several morphological nuances on the nouns influence the actual head word of the noun. For example,

```
Sendra varam janaNayagak katchi nirvagigal oliraKonth.
```

In this sentence, all of the words have been assigned a part of speech tag of 'NN'. In many such compound noun expressions, the parser makes mistakes. I think if we had more morphological features that could characterize these nouns, the parser can better judge head relations. Also, another common error I found was the head being wrongly labeled. In Tamil, in most of the sentences - the verb at the final clause usually seemed to be the head. But in some cases, the auxiliary verb could also act as the head. Unfortunately, the parser made mistakes in this decision and by choosing the wrong head - a lot of dependencies involving a word to the actual head were marked incorrectly. Following are a list of experiments, I want to try out in the next phase.

1. Annotate more training data of about 5000 tokens.
2. Add morphological features to MALT Parser.

8 System Analysis on Corpus B

For this round of evaluation, I managed to annotate around 4000 tokens for training. I tried to follow the annotation methodology that I had followed during my earlier annotation rounds and tried to improve my training set. In the previous section, I had mentioned that I wanted to use morphological parses as additional features for supervised training. But the resource that I had planned to use was not reliable `http://ltrc.iiit.ac.in/showfile.php?filename=downloads/shallow_parser.php` and I was not able to produce valid parses for several of the words. For example, at-least 6/7 of the tokens did not have any parse output. Apart from this tool, I could not find any other usable resource that worked. For this week's analysis - I used standard MaltParser settings to train on the 4000 tokens that I had annotated and tested it on my corpus B. I got an accuracy of about **66%**. The performance was much better than what I had expected. I think that

Corpus	Model	Accuracy (%)
Corpora A	Basic setting	68.3
Corpora B	Basic setting	67

Table 1: 4350 tokens + Standard MaltParser settings

additional training data definitely helped in this case. I did a brief analysis of the results of my earlier model compared with my current model and one major improvement that I could see was in identifying longer head relationships. In my previous model, there were a lot of errors when a word had a head much farther in the clause. These type of errors had decreased a lot in my new model. Also, the sentences where the 'head' was incorrectly identified in my previous model had reduced. Also, in my previous model - the number of errors on larger sentences was very high. My newer model performed much better on longer sentences which was encouraging because I had quite a large number of sentences with more than 20 tokens. I also tried another experiment where I applied the same model that I had built on the same training set. I got an accuracy of around 82%. This was encouraging because, it means that the rules that have been learned during training fairly distribute well throughout the training corpus. Since, I was not able to extract Morphological features, I am planning to use additional unsupervised features that can help capture context of different words and could also help in handling unseen words. I am currently experimenting with Brown clustering and to generate clusters that can be incorporated into Malt Parser's feature representation.

9 Final Revisions

9.1 Evaluation on Corpus A and B

I found some inconsistencies in my annotations when abbreviations were involved. For example, when I had abbreviations - the tokenizer that I used split based on punctuations and my tokenizer generated multiple tokens for a single word. For example, for the token '23.May-1949', the tokenizer generated multiple tokens. I decided to attach all the tokens that were split by the tokenizer to the last token. In this case, '23' and 'May' would have 1949 as the head. I also performed some manual cleaning up of the training data and removed those sentences for which the POS tagger performed very inaccurately. My final training corpus consisted of around 4350 tokens. My baseline results on both Corpus A and Corpus B using standard MaltParser settings are given in Table 1.

As we can see, the results across both the Corpora are somewhat similar. Both Corpora A and B consist of news articles and this shows that the model that was trained performs somewhat similar across both corpora.

9.2 Experiments with different learning algorithms in MALT Parser

MaltParser comes with three families of parsing algorithms (Nivre, Stack - Nivre (2009)Nivre et al. (2009) and Covington - Covington (2001)). I decided to experiment with all these algorithms for training and my results are described in Table 2 and 3.

The default Nivre Arc Eager algorithm performs the best while the other algorithms perform worse than the baseline in both corpora. Also, the arc Eager parser outperforms the lazy and projective settings in all the parsing algorithms. In Tamil, most of the dependency structures are projective in nature. But there are many instances where certain structures are non projective. The arc eager parser can handle non projective structures and outperforms the projective mode which is mostly suited for projective dependency structures. Even though, the lazy mode can handle

Malt Parser Model	Accuracy (%)
Default Setting (Arc Eager)	**68.3**
NivreEager Arc Standard	67
Covington	67.7
Stack Projective	67.25
Stack Eager	67.7
Stack Lazy	67

Table 2: Corpus A results for various MaltParser algorithms

Malt Parser Model	Accuracy (%)
Default Setting (Arc Eager)	**67**
NivreEager Arc Standard	64.8
Covington	66.7
Stack Projective	64.3
Stack Eager	64.7
Stack Lazy	64

Table 3: Corpus B results for various MaltParser algorithms

non-projective structure - the lazy version applies the non-projective transition only at the very end. This could sometimes result in non-projective dependencies not being captured.

9.3 Adding Morphological features for training

I had earlier mentioned in the previous sections that I wanted to incorporate morphological features in the parsing model. The shallow parser tool (IIIT Hyderabad (2013)) that I used, did not work locally for me when I installed the software on my system. But, there was an online demo of the shallow parser's capabilities that took sentence as an input in a form and produced morphological annotations for each lexical item. I had to write a script to scrape contents from the website and extract the morphological features from the website. The output was in Shakti standard format (Bharati et al. (2007)), which is a format for describing the chunk relations that exist in the sentence and also the morphological parse for each lexical item. The output of the morphological analyzer for each lexical item is a comma separated string consisting of 7 slots. The slot values are **root, category, gender, number, person, case** of a lexical item. The crawl process was very slow and I had to deal with a lot of formatting issues when fetching the information from the website. For unknown, words - an 'unk' tag is assigned.

The results are discussed in Table 4. As we can see, adding morphological features definitely helped in improving the performance of the parser. In both the corpora, the performance improved and I was able to get a **1**% improvement on corpora A while the improvement in corpora B was slightly less. Although, I had a lot of unknown tags in the extracted output, the results indicate that morphological features are very useful for a highly inflected language like Tamil. Among the morphological features, the lemma of the word proved most useful and provided the greatest improvement. The other slots were very sparse and contained a lot of 'unk' tag - which might indicate why other information such as tense, gender and case were not very useful.

Corpora	Morph information	Accuracy (%)
Corpora A	Only lemma	69.2
Corpora B	Only lemma	67.7
Corpora A	lemma + other morph features	**69.25**
Corpora B	lemma + other morph features	**67.7**

Table 4: Using morphological features for training

Model	Accuracy (%)
Clusters(N=100)	68.6
Clusters(N=50)	68.8
Clusters(N=100) + morph features	69.3
Clusters(N=50) + morph features	69.7

Table 5: Incorporating brown clusters (Corpora A)

9.4 Unsupervised learning using Brown clustering

Since, I had a large number of unannotated data in my corpora, I decided to use these sentences to obtain clusters for the various words in the corpora. My input data for clustering consisted of a total of 2727174 tokens. The clusters would be generated based on the contexts on the words, and words that appear in similar contexts would have similar clusters. I used Percy Liang's implementation (Liang (2005) , Brown et al. (1992)) of the Brown clustering algorithm and experimenting by using 100 and 50 clusters. The clustering output consists of bit strings and I used them directly as features in my MaltParser model. I ran experiments using both the basic Part of Speech model and also by incorporating Morph features. The results are given in Table - 5 and 6.

As I expected, clustering features improved performance across both the Corpora. Since all of the data comes from a particular set of domains, the effect of clustering should reflect similarly across both the corpora. However, the improvement in performance was not that significant and I think that one reason is because the number of unique tokens was very high in the corpora. Tamil has a lot of inflected variants of several nouns and verbs and not all of these words occur frequently in the corpus. The cluster assignments for these words might not be of high confidence, which might have resulted in the relatively trivial improvement in performance.

Model	Accuracy (%)
Clusters(N=100)	67.1
Clusters(N=50)	67.3
Clusters(N=100) + morph features	67.7
Clusters(N=50) + morph features	67.9

Table 6: Incorporating brown clusters (Corpora B)

10 Conclusion and Future work

The best unlabeled dependency attachment score that I was able to achieve was about 69.7% for Corpus A and 67.9% for Corpus B. Using both morphological features and cluster labels obtained by Brown clustering certainly improves the performance of the parser. But, the training data that I have developed for this work is still very small. More training data is needed to build industry standard parsers that can achieve high accuracies. One of the major setbacks that I had in this project was the lack of availability of good industry standard tokenization and morphological analysis tools. Tokenizers that could split on various clitics like 'um', 'aaga' could certainly improve the quality of tokens produced for Tamil. For future work, I want to experiment with chunk features described in Ambati et al. (2010). Local syntactic features can certainly help in capturing relationships among tokens and I think this would certainly help in improving the performance of the parsing algorithm. Also, I want to experiment by clustering the lemmas of all tokens using Brown clustering and using that cluster information as a feature for training. I think this approach would be able to handle relatively sparse tokens and we would be able to get more meaningful clusters.

Also the training corpora that I created was solely done by me and it might consist of several mistakes. Peer validation of the training data might help in creating a better and more consistent corpora. Another approach that might work is the incorporation of several manual rules as features to the model. For example, in Tamil - the attachment of many tokens to their corresponding head depends on a lot of factors such as postpositions, inflections etc. Such phenomena can be better captured using rule based features.

References

Bharat Ram Ambati, Samar Husain, Sambhav Jain, Dipti Misra Sharma, and Rajeev Sangal. Two methods to incorporate local morphosyntactic features in hindi dependency parsing. In *Proceedings of the NAACL HLT 2010 First Workshop on Statistical Parsing of Morphologically-Rich Languages*, pages 22–30. Association for Computational Linguistics, 2010.

Akshar Bharati, Rajeev Sangal, and Dipti M Sharma. Ssf: Shakti standard format guide. *Language Technologies Research Centre, International Institute of Information Technology, Hyderabad, India*, pages 1–25, 2007.

Alena Böhmová, Jan Hajič, Eva Hajičová, and Barbora Hladká. The prague dependency treebank. In *Treebanks*, pages 103–127. Springer, 2003.

Peter F Brown, Peter V Desouza, Robert L Mercer, Vincent J Della Pietra, and Jenifer C Lai. Class-based n-gram models of natural language. *Computational linguistics*, 18(4):467–479, 1992.

Michael A Covington. A fundamental algorithm for dependency parsing. In *Proceedings of the 39th annual ACM southeast conference*, pages 95–102. Citeseer, 2001.

V Dhanalakshmi and S Rajendran. Natural language processing tools for tamil grammar learning and teaching. *International journal of Computer Applications (0975-8887)*, 8(14), 2010.

LTRC IIIT Hyderabad. Shallow parser for Tamil. `http://ltrc.iiit.ac.in/showfile.php?filename=downloads/shallow_parser.php`, 2013.

T. Lehmann, Pondicherry Institute of Linguistics, and Culture. *A Grammar of Modern Tamil.* PILC publication. Pondicherry Institute of Linguistics and Culture, 1989. URL `http://books.google.com/books?id=THlkAAAAMAAJ`.

Percy Liang. *Semi-supervised learning for natural language.* PhD thesis, Massachusetts Institute of Technology, 2005.

Joakim Nivre. Non-projective dependency parsing in expected linear time. 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 1-Volume 1, pages 351–359. Association for Computational Linguistics, 2009.

Joakim Nivre, Marco Kuhlmann, and Johan Hall. An improved oracle for dependency parsing with online reordering. In *Proceedings of the 11th international conference on parsing technologies*, pages 73–76. Association for Computational Linguistics, 2009.

S Rajendran. Tamil wordnet.

Loganathan Ramasamy and Zdeněk Žabokrtskỳ. Tamil dependency parsing: results using rule based and corpus based approaches. In *Computational Linguistics and Intelligent Text Processing*, pages 82–95. Springer, 2011.

Loganathan Ramasamy, Ondřej Bojar, and Zdeněk Žabokrtský. Tamil dependency treebank (tamiltb) 0.1. annotation manual. Technical Report TR-2011-42, Institute of Formal and Applied Linguistics (FAL MFF UK), Faculty of Mathematics and Physics, Charles University, 2011.

Loganathan Ramasamy, Ondřej Bojar, and Zdeněk Žabokrtský. Morphological processing for english-tamil statistical machine translation. In *Proceedings of the Workshop on Machine Translation and Parsing in Indian Languages (MTPIL-2012)*, pages 113–122, 2012.

Loganathan R Santhosh Kumar C Srinivasan CJ, Udhaykumar N. Robust dependency parser for natural language dialog systems in tamil. *5th Workshop on Knowledge and Reasoning in Practical Dialogue Systems*, pages 1–6, 2007.

Wikipedia. Tamil language. 2013. URL `http://en.wikipedia.org/wiki/Tamil_language`.