This paper describes a two-phase Turkish dependency parsing which separates dependency and labeling into two similar to (McDonald et al., 2006b). First, in order to solve the long distance dependency attachment problem, the sentences are split into constituents and the dependencies are estimated on shorter sentences. Later, for better estimation of labels, Conditional Random Fields (CRFs) are used with previously learned chunk and several dependency and morphosyntactic features. Finally, a post-processing step is applied to “correct” some of labels, if necessary.

1 Introduction

Dependency parsing, a well-studied problem in natural language processing, is the task of forming a dependency tree by attaching each word of a sentence (dependent) to another word in the same sentence (head) with a label that describes the dependency relation between these words. In the last decade, the data-driven dependency parsing approaches (Nivre et al., 2007; McDonald et al., 2006a) have received a considerable attention as it learns solely from labeled data and can be rapidly adapted to new languages and domains.

The accuracy of a dependency parser is negatively affected by two factors, among possibly others. First, a parser’s accuracy is sensitive to sentence length (McDonald and Nivre, 2007). As the parsers tend to assign dependencies in relatively short distances (Nivre et al., 2006), long sentences are not easy to parse correctly. Second, wrong labels that are assigned to correct dependencies result in labeled accuracy drop.

In data-driven dependency parsing approaches (Nivre et al., 2007; McDonald et al., 2006a), the dependencies and labels are often learned at the same time. To our best knowledge, the work by McDonald et al. (2006b) is unique in that it learns the dependencies and labels in two separate stages. In this paper, we present a two-phase data-driven dependency parsing of Turkish that addresses the above-mentioned problems in consecutive steps.

In order to solve the long distance dependency attachment problem, we first split sentences into their constituents. For each constituent, we construct a sub-sentence by appending the verb group of the original sentence to the end of the constituent. We then parse all these short sentences by the MaltParser (Nivre et al., 2007) which is trained with Turkish specific parameters (Eryiğit et al., 2008). Finally, we combine the generated sentences to form the original sentence with full dependencies.

For the labeling problem, we use a CRF-based (Lafferty et al., 2001) approach with the use of chunk information and parser output for identifying dependencies. On top of our CRF-based labeling approach, we also apply a post-processing step to correct dependency labels if necessary. Our methodology improves the state-of-the-art Turkish dependency parsing (Eryiğit et al., 2008) with a 1.7% increase in the labeled attachment score \( AS_L \) and 0.4% increase in the unlabeled attachment score \( AS_U \).

There are several related research on incorporating different features during the parsing such as chunk (Attardi and Dell’Orletta, 2008) and causal (Gadde et al., 2010) and morphosyntactic features (Ambati et al., 2010). Our works differs from several aspects; first, instead of using the chunk information as a...
feature in the parsing, we used the chunk information as a preprocessing step to split the sentences into “shorter” ones and in the second step of parsing while estimating the labels. Second, as an addition to the word’s morphosyntactic features, we employed the morphosyntactic features of the head word for each token again in label estimation step.

2 Turkish Dependency Parsing

Turkish, a member of the Turkic languages, is an agglutinative language with very productive inflectional and derivational morphology. From the dependency point of view, Turkish is generally a head-final language. The dependency relation arcs are formed from left to right and do not cross each other except in some rare cases (Oflazer et al., 2003). The first investigations on a Turkish dependency parser was done by Oflazer (Oflazer, 2003). Following this grammar-based work, Turkish dependency parsing was investigated by Eryiğit et al. (Eryiğit et al., 2008) where the impact of morphological units on different types of parsers was explored. This study showed that the correct dependency representation of a Turkish sentence should use root words and inflectional groups (IGs)\(^1\) instead of the whole words themselves. The best performing Turkish dependency parsing is obtained via the data-driven MaltParser (Nivre et al., 2007) by using Turkish-specific features.

For training the parser, we used the Turkish dependency treebank (Oflazer et al., 2003) that is also used in CoNLL-X (Buchholz and Marsi, 2006)\(^2\). The treebank corpus contains several features including word, lemma, POS, coarse POS\(^3\), and IGs that reflect the morphological structure of Turkish.

Recently, Turkish dependency parsing was improved with the use of multiword expressions (Eryiğit et al., 2011) and the effects of automatic morphological analyzer and disambiguation were explored (Eryiğit, 2012).

To show that the transition-based dependency parsers are in favor of identifying dependencies in short distances, we examined the dependency attachments of the MaltParser with respect to distance on the ITU validation set (Eryiğit, 2007)\(^4\). We classified the dependency attachments into three categories with respect to the number of words that occur between the attached words: i) Short, dependency attachments at a distance of 1 or 2 words, ii) Mid, dependency attachments at a distance between 3 and 6 words, and iii) Long, dependency attachments at a distance of 6 or more words. Table 2 shows the comparison of the distances in the gold data and in the attachments identified by the dependency parser for the validation set (all/correct attachments). As can be seen from the results, the Turkish dependency parser assigns 66% of the “correct” long distance attachments and 58% of the mid distance attachments that appear in the gold data.

3 Incorporating Constituents as a Preprocessing Step

A constituent is a group of words that behave as a single unit from the structural and meaning views. Constituents can be omitted in the sentence without influencing the sentence grammaticality. Constituents are the word groups that we can be found by asking the questions “who/what”, “when”, “where”, etc. to the verb constituent. Most frequent Turkish constituents can be listed as Subject, Sentence (Verb), Object, and Adjunct. The main constituent of a sentence is its verb and other constituents can form a

\(^1\)To represent the morphology, words are separated at the derivational boundaries. The inflectional morphemes with derivation (or the root word for the first IG) are called as inflectional groups.

\(^2\)Shared Task on Multilingual Dependency Parsing

\(^3\)The Turkish morphological analyzer gives a two-layered POS information such as Noun+Proper and Num+Ordinal. The coarse POS is the first part of this POS information. In the absence of the second layer, the POS is the coarse POS.

\(^4\)This set was used as the test set in this work.
sentence with the presence of the verb. Every word in a constituent is dependent to a word within the constituent except the head word of the constituent. Head word of the constituent is dependent to the verb. Any constituent becomes an ungrammatical and meaningless structure if its head word is removed.

Figure 1 illustrates the chunking process for the sentence “Öteleme işleminde kuyruktaki bekleyen eleman yiğına itilir” (The element in the queue is pushed onto the stack during the shifting process.). This sentence contains four constituents sequentially, Locative.Adjunct, Subject, Dative.Adjunct and finally (Sentence/Verb). One can observe by dropping the head word “isleminde” in the Locative.Adjunct constituent, constituent loses its meaning completely. Each of these constituents (except the verb) are “phrases” alone and can form a sentence only with the verb chunk. After dropping any constituent (again except the verb), for example the Subject, the sentence is still a grammatical Turkish sentence as “Öteleme yiğına itilir” (is pushed onto the stack during the shifting process.)

In our work, we used the Turkish dependency treebank and ITU validation test set which is enriched with the chunk information (Durgar El-Kahlout and Akın, 2013). The chunker is reported to work with an F-measure 91.95 for verb chunks and 87.50 for the other chunks. In that work, only verb chunks are labeled separately and the other chunks are labeled with the same type such as [CHUNK Öteleme işleminde] [CHUNK kuyruktaki bekleyen eleman] [CHUNK yiğına] [VERB itilir].

3.1 Procedure

Before the parsing process, we split each sentence of the test set into their constituents. The idea behind splitting sentences was to create synthetically shorter sentences in order to make the dependency parsing task easier by “shortening” long distance dependencies. For each constituent, we generated a sub-sentence by appending the verb to the end of the constituent. As a result, we generated $n - 1$ new sentences from a sentence with $n$ constituents. Part (b) of Figure 1 illustrates the generation of shorter sentences from the chunked sentence shown in part (a) of the same figure. Each of these shorter sentences are grammatical for Turkish. The shorter sentences contains only the dependencies within the constituent and the dependencies of the constituent to the verb constituent.

After splitting the original sentence into a number of shorter sentences, each of these sentences are parsed by the MaltParser in order to obtain the dependencies. Finally, these parsed sentences were combined into a whole in such a way that the original sentence was generated back with identified dependency relations.

Splitting complex sentences with more than one verb group was not trivial. We classified these sentences into two groups; complex sentences with two verb groups and complex sentences with more than two verb groups. For the first case, the last verb group was considered as the dominating one and the sentence was split accordingly to this verb group. In this work, we didn’t split sentences that belong to the second group and kept them as whole sentences in our experiments.

Figure 2 illustrates a sample Turkish sentence, its dependency parse generated by the MaltParser (part (a)), and the gold parse (part (b)) of the sentence. As can be seen from this example, the parser mistakenly attaches the word “işleminde” to the word “bekleyen” (shown with a dotted link in part (a)). The figure also shows the parses of the shorter sentences generated from this sentence (part (c)). The generation of the original sentence from the parses of shorter sentences (part (d)) are also given in the figure. It is noteworthy to mention that splitting the original sentence and parsing shorter sentences individually enables the parser to find the correct attachment of the word “işleminde” to the verb “itilir” (as is in the
gold output part (b)).

3.2 Results

For our evaluations, we used the evaluation tool distributed with the MaltParser. The performance of a dependency parser is mainly evaluated with three scores: the labeled attachment score ($AS_L$); the unlabeled attachment score ($AS_U$) and the label accuracy score ($LA$). We conducted experiments both on the chunked sentences using Turkish constituent chunker (Durgar El-Kahlout and Akın, 2013) and gold chunks as described in Section 3.1.

Table 2 compares the precision and recall scores of the MaltParser output with original sentences and our approach relative to the dependency attachment distance. The results showed that dependency parsing with the use of constituent chunks increased the recall for all distance lengths (i.e., up to 6 points) and improved the precision approximately 3 points for dependency distances between 3 and 6 words and more than 13 points for dependency distances more than 6 words.

To see the effects of sentence lengths on parsing performance, we split sentences relative to their lengths (i.e., 1-8, 9-15, 16-30, and >30) and reported the scores with respect to different length groups. Table 3 shows the parser performance depending on sentence lengths. The results showed that $AS_U$ was improved up to 1.5 points for all sentence lengths. For shorter sentences, the $AS_L$ was relatively worse than the parses of original sentences but for longer sentences the performance was better with the gold chunks. For the chunker output, performance slightly better for sentences with 16 to 30 words for the chunker chunks. It is particularly noteworthy to mention that the labeling (the label accuracy result) was worse than the parses of original sentences for all sentence lengths. Our approach in a second step
improved the label accuracy as described in Section 4.

4 Relabeling the parser output

In the parses of original sentences, we observed that the intersection of the correct dependencies (2461) and the correct labels (2565) is only 2216 out of 3080 attachments. This shows us that approximately 10% of the correct dependencies are missed because of the wrong labels; this causes an accuracy drop in the ASL score.

Assigning dependency labels can be approximated as a sequential labeling task for Turkish with the projectivity assumption, where the dependency tags are associated with every token in a sequence of tokens (Ramshaw and Marcus, 1995). Adding features of the head word’s (that is learnt in the previous step) can be included to each token to create syntetically sequential data.

To assign the labels, we used CRFs (Lafferty et al., 2001) which became the state-of-the-art framework for several labeling tasks such as text segmentation, named entity recognition, part of speech tagging, and shallow parsing. They are shown to outperform the probabilistic models such as HMMs (Church, 1988; Freitag and McCallum, 2000) and MEMMs (McCallum et al., 2000) in several sequential assignment tasks.

4.1 Features

To model the label attachment problem with CRFs, we used four types of features; i) baseline features: the set of features that exists in the Turkish dependency treebank, ii) morphological features: the features that are split from the IG information, iii) dependency features: features that are extracted from the first phase of the dependency parsing, and iv) chunk features: the features from the chunk annotation. The full set of features that are used in the dependency labeling task are as follows:

- **Baseline Features**: Word, Lemma, The main POS of the word (CPOS), The second layer of the POS information of the word (POS), The combined inflectional morpheme information of the word’s last inflectional group (IG).

- **Morphological Features**: The case of the word when its POS is Noun or Pronoun (CASE). The feature can take the values Acc, Dat, Nom, Loc, or Abl.

- **Dependency Features**: The word’s distance to its head (DIST); If the word is attached to a head within a distance of one or two words then the distance is short, otherwise it is long. Head word CPOS (HCPOS), lemma of the word’s head (HLEM).

- **Chunk Features**: Chunk type (ChnkTYPE), Chunk type is Verb for the sentence/verb chunks and Regular for the rest of the chunks. The chunk type is Undefined if the token is not assigned to any chunk.

---

5The situation is more or less same in the output of our approach.

6This excludes the derivation and punctuation tokens.

7To represent the morphology, words are separated at the derivational boundaries. The inflectional morphemes with derivation (or the root word for the first IG) are called as inflectional groups.

8The Case information also exists in the IG feature but combined with the Person and Number information.

Table 3: Evaluation Relative to the Sentence Lengths.
To the train the CRF relabeling, the gold labels (morphology, dependencies, etc.) are used for each type of features. Table 4 shows the complete set of features used for the Turkish sentence “Burada çiçeklerin satıldığı geniş bir alan vardı” (There used to be a huge area here where the flowers were sold).

### 4.2 Post-processing

To better estimate the labels, we should figure out the type of labeling errors that the dependency parser produces. In order to designate such kind of errors, we parsed 50 sentences from a Turkish corpus (different from the test set). After a manual inspection, we observed that from several others, some of the “DATIVE.ADJUNCT”s are labeled as “OBJECT”. This error can be easily corrected by just controlling the Case feature of the token. In Turkish, specific adjuncts ends with specific case suffixes. For example, all dative adjuncts have the Case “Dat”. Both MaltParser and our labeling procedure fails to assign the correct label for this case. So, after the relabeling procedure, we replaced every token with the label “OBJECT” and the Case feature “Dat” to “DATIVE.ADJUNCT”. This manual post-processing corrected 41 (out of 56) of the problematic cases on the test set.

### 4.3 Results

We used the CRF++ tool, to train and test the Turkish dependency labeling. The window size of the sentence was 5 taking the preceding two words and following two words. For training, we used all Turkish dependency treebank data. As the test data, we used the output produced in the first phase of parsing. Table 5 compares the performance of the labeling according to the $\text{AL$_L$}$ and $\text{AL$_A$}$ scores for original sentences, attachments obtained by the gold chunks and chunker chunks in the previous step. As a result, we obtained same $\text{AL$_L$}$ without the post-processing step score and 0.26 points in $\text{AL$_L$}$ after the post-processing step with better $\text{AL$_A$}$ scores over the MaltParser output of the original sentences with the chunker output. The performance is much better with the gold chunked assignments. Our experiments showed that using a CRF-based labeling enhanced with extra features increased the label accuracy and outperformed the Turkish dependency parser with respect to $\text{AL$_L$}$.

### 5 Results and Main Findings

Dependency parsers have problems in assigning long distance dependencies. They tend to assign dependencies relatively in short distances. In order to solve this problem, we offered a new chunking-based parsing methodology. Our methodology first chunks sentences into constituents. For each constituent, one grammatically correct sentence is generated with some meaning loss by attaching the verb chunk and parsed with MaltParser. First chunking a sentence and then using the dependency parser outperforms the state-of-the-art results. However, the results with respect to the $\text{AL$_L$}$ is not satisfying due to wrong labeling. Thus, in a second phase, our approach treats labeling as a sequential attachment problem. CRF-based models are used with enhanced features extracted from morphological structure, chunks and dependency attachments.

In our experiments, 52 sentences out of 300 sentences were not split as either they have only one chunk or more than two verb chunks. We generated approximately 2.69 short sentences from each of

---

9CRF++: Yet Another CRF toolkit.
<table>
<thead>
<tr>
<th>Method</th>
<th>( AS_L )</th>
<th>LA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original Sents.</td>
<td>71.95</td>
<td>83.28</td>
</tr>
<tr>
<td>+POST</td>
<td>72.95</td>
<td>84.51</td>
</tr>
<tr>
<td><strong>Chunks - Gold</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Shortened Sents.</td>
<td>72.05</td>
<td>82.37</td>
</tr>
<tr>
<td>+POST</td>
<td>73.28</td>
<td>83.67</td>
</tr>
<tr>
<td>CRF Feat.s</td>
<td>72.63</td>
<td>83.21</td>
</tr>
<tr>
<td>+POST</td>
<td><strong>73.86</strong></td>
<td><strong>84.51</strong></td>
</tr>
<tr>
<td><strong>Chunks - Chunker</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Shortened Sents.</td>
<td>71.40</td>
<td>82.44</td>
</tr>
<tr>
<td>+POST</td>
<td>72.56</td>
<td>83.73</td>
</tr>
<tr>
<td>CRF Feat.s</td>
<td>71.95</td>
<td>83.51</td>
</tr>
<tr>
<td>+POST</td>
<td><strong>73.21</strong></td>
<td><strong>84.81</strong></td>
</tr>
</tbody>
</table>

Table 5: CRF-based Labeling and Post-processing Results.

the remaining sentence. The performance with respect to \( AS_L \) was improved from 71.95 to 73.21 (an improvement of 1.7%) and with respect to \( AS_U \) from 79.90 to 80.23 (an improvement of 0.4%) over parses of original sentences.

6 Conclusions and Future Work

In this paper, we presented a two-phase Turkish dependency parsing approach where the dependencies are identified in the first phase and the labels are identified in the second phase. We improved dependency attachments by chunking sentences into their constituents, generating shorter sentences from these constituents, finding dependencies in these shorter sentence, and finally generating the original sentence back from these dependency parses. For the labeling task, we used a CRF-based approach enriched with extra features from the morphological information, dependencies and chunking. Moreover, we performed a rule-based post-processing to correct some dependency labels, if necessary.

Future work includes splitting the dependency treebank into constituents also and train the parser with shorter sentences similar to the test data. Because the lack of different test sets for Turkish, we will also make 10-fold cross validation with the training data. Moreover, we are planning to replicate the experiments with different state-of-the-art parsers such as Bohnet parser (Bohnet and Nivre, 2012).

References


