Chunking in Turkish with Conditional Random Fields

Olcay Taner Yıldız[1], Ercan Solak[1], Razieh Ehsani[1], and Onur Görgün[1,2]

[1] Işık University, Istanbul, Turkey
[2] Alcatel Lucent Teletaş Telekomünikasyon A.Ş., Istanbul, Turkey

Abstract. In this paper, we report our work on chunking in Turkish. We used the data that we generated by manually translating a subset of the Penn Treebank. We exploited the already available tags in the trees to automatically identify and label chunks in their Turkish translations. We used conditional random fields (CRF) to train a model over the annotated data. We report our results on different levels of chunk resolution.

1 Introduction

Chunking is one of the earlier steps of parsing and defined as dividing a sentence into syntactically disjoint sets of meaningful word-groups or chunks. The concept of phrase chunking was proposed by [1]. [1] argued that words could be brought together into disjoint 'chunks' and therefore on the whole simpler phrases in general. An example of a sentence split up into chunks is shown below:

(1) [NP John] [VP guesses] [NP the current deficit] [VP will decrease] [PP to only $1.3 billion] [PP in October]

The chunks are represented as groups of words between square brackets, and the tags denote the type of the chunk.

Extracted chunks can be used as input in more complex natural language processing tasks, such as information retrieval, document summarization, question answering, statistical machine translation, etc. Compared to syntactic and/or dependency parsing, chunking is an easier problem. For this reason, in the earlier years of statistical natural language processing, many researchers put emphasis on the chunking problem [2].

[3] was one of the earliest works. Their transformation-based learning approach achieved over 90% accuracy for NP chunks on the data derived from Penn-Treebank. [4] applied support vector machines (SVM) to identify NP chunks. Using an ensemble of 8 SVM-based systems, they got 93% in terms of F-measure. [5] applied generalized winnow (a classifier much simpler than the ensembles of SVM's) on the overall chunking problem, and get an average of 94% in terms of F-measure on CoNLL shared task data [6]. [6] give an overview of the CoNLL shared task of chunking. 10 different chunk types covered in CoNLL shared task are ADJP, ADVP, CONJP, INTJ, LST, NP, PP, PRT, SBAR, and VP. Training material consists of the 9,000 Wall Street Journal sentences augmented with POS tags. Although there is quite a bit of work in English chunking done in the last two decades, the work in other languages, especially on less resourced and/or morphologically rich languages, is scarce. There are limited works on Korean [7,8], Hindi [9], and Chinese [10,11].

A. Gelbukh (Ed.): CICLing 2015, Part I, LNCS 9041, pp. 173–184, 2015.
DOI: 10.1007/978-3-319-18111-0_14

[12] implemented the first Turkish NP chunker which uses dependency parser with handcrafted rules. NP's are divided into two sub-classes as main NPs (base NPs) and all NPs (including sub-NPs). Noun phrases with relative clauses are omitted in his work. [13] introduced a new chunk definition by replacing phrase chunks with constituent chunks. They used METU-Sabancı Turkish dependency treebank [14] for chunk annotation and only labeled verb chunks. The remaining chunks are left as general chunks and only their boundaries are detected. The algorithm is based on conditional random fields (CRF) enhanced with morphological and contextual features. According to the experiment results, their CRF-based chunker achieves a best accuracy (in terms of F-measure) of 91.95 for verb chunks and 87.50 for general chunks.

In this paper, we propose a general CRF-based Turkish chunker. Our contributions are two-fold; (i) we automatically construct a chunking corpus using the parallel Turkish treebank of about 5K sentences translated out of Penn-Treebank [15] (ii) we improve upon the work of [13] by learning all common chunk types in Penn-Treebank. We define three learning problems in increasing levels of difficulty. In the first level, we only identify the boundaries of chunks. In the second level, we detect chunk types. In the third level, we try to discriminate chunk sub-types (NP-SBJ, NP-OBJ, ADVP-TMP, etc.) of the chunks in the second level.

The paper is organized as follows: In Section 2, we give a very brief overview on Turkish syntax. We give the details of our data preparation steps in Section 3 and describe the CRF for chunking in Section 5. Our CRF features for chunking are detailed in Section 6 and we give the experimental results using those features in Section 7. We conclude in Section 8.

2 Turkish

Turkish is an agglutinative language with rich derivational and inflectional morphology. Morphemes attach to the stems as suffixes. Word forms usually have a complex yet fairly regular morphotactics. Most suffix morphemes have several allophones which are selected by morphophonemic constraints, [16].

Turkish sentences have an unmarked SOV constituent order. However, depending on the discourse, constituents are often scrambled to emphasize, topicalize, focus and background certain elements. Writing and formal speech tend towards the unmarked order.

Turkish is a head final language. Adjectives qualifying a head in a noun phrase are usually scrambled for emphasis. Even then, intonation in speech is used to add a further layer of emphasis.

Below is a Turkish sentence with the chunks identified in brackets and subscript labels.

(2) [Dün]$_{\mathrm{ADJP}}$ [büyük bir araba]$_{\mathrm{NP}}$ [beyaz binanın önüne]$_{\mathrm{PP}}$ [geldi]$_{\mathrm{VP}}$.
[White building.GEN front.DAT]$_{\mathrm{PP}}$ [yesterday]$_{\mathrm{ADJP}}$ [big a car]$_{\mathrm{NP}}$ [come.PAST.3SG]$_{\mathrm{VP}}$.
Yesterday, a big car came to the front of the white building.

In Turkish syntax, case markings of the heads identify the syntactic functions of their constituents. For example, accusative marking identifies the direct object of a transitive

verb. Similarly, dative marker sometimes denotes the directional aspect of its phrase and sometimes marks the oblique object of a verb. Although this aspect of Turkish syntax makes it a bit peculiar, in this work, we refrained from inventing our own set of Turkish chunk labels and used the same set of chunk labels that are generally used in English chunking.

3 Data Preparation

Constructing a chunking corpus from scratch by manually tagging and chunking linguistic data is very expensive. Instead, one can use already tagged corpora to automatically generate chunking corpus. Penn Treebank is a fairly large collection of English sentences annotated with their detailed constituent parses. In this work, we used the Penn Treebank and generated chunked sentences automatically. Below we explain the detailed corpus generation process.

For our previous SMT work, we generated a parallel corpus by manually translating a subset of English sentences in Penn Treebank to Turkish [17]. In the chunking task, we used this corpus to automatically generate chunks in Turkish sentences.

Throughout the manual translation, we had used the following constraints. We kept the same set of English tags. We did not introduce new tags either for POS labels or phrase categories. Furthermore, we constrained our translated Turkish sentences so that trees for Turkish and English sentences are permutations of each other. A human translator starts with an English tree and permutes the subtrees of the nodes recursively as necessary until he arrives at an acceptable word order for a Turkish sentence. Finally, he replaces the English word tokens at the leaves with Turkish tokens or the special *NONE* token that denotes a deletion. The Figure 1 illustrates the process between the parallel sentences in (2)

Next, we describe the steps involved in processing a Turkish parse tree to generate chunking data. We used the 8 labels in Table 1 to identify the basic categories of chunks.

Table 1. Chunk labels

Chunk label	Description
ADJP	Adjective phrase
ADVP	Adverb phrase
NP	Noun phrase
PP	Prepositional phrase
S	General clause
CC	Coordinating conj
PP	Prepositional phrase
VG	Verb group
PUP	Punctuation

Given the parse tree of a Turkish sentence, we traverse it breadth first. For each node except the root encountered in the traversal, if the node tag is in the Table 1, we do not

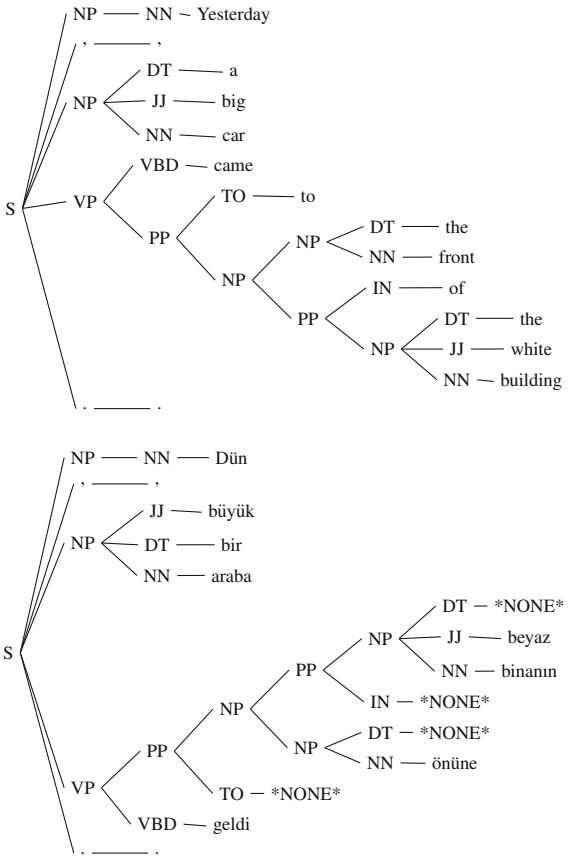

Fig. 1. The permutation of the nodes and the replacement of the leaves by the glosses or *NONE*

traverse its children further. We chunk the tokens in its leaves and label them with the node tag.

The last label VG in Table 1 is generated using the verb phrase VP with the some modifications to accommodate Turkish verb structure. In the Penn Treebank, VP often groups together the object NP and several PP and ADVP phrases under the same tree tagged as VP. Thus, all of these constituents would be put together as a VP chunk. However, for chunking, such a grouping is too coarse. We did not use VP as a chunk in our data. Instead, we first automatically identified and extracted the verb under VP and chunked it as a verb group, VG. Then, we identified the remaining subtrees of VP tree as chunks with their own labels by traversing it breadth first as above. For example, for the tree in Figure 1, we extract "geldi" as the verb group and extract out of VP subtree the phrase "beyaz binanın önüne" as PP.

In the Penn Treebank, syntactic tag set includes several distinct tags, SBAR, SBARQ, SINV, SQ, for subordinate clauses, question and inversion constituents etc. For chunking, we lumped these categories under a single category S.

We had about 5K sentences to begin with. Many of those are not full sentences but fragments. In generating the chunking corpus, we used only the full sentences. This reduced our corpus to about 4K sentences. We used this reduced set of Turkish parse trees to generate chunking data for training and testing. We confined our selection so that our training set is derived from the training subset of the Penn Treebank. For our test set, we combined the development and the test subsets derived from the Penn Treebank. As a result, we generated 3681 training sentences and 723 test sentences for our chunking task.

4 Chunking Levels

We treat the chunking problem as one of sequence labeling. We divide the labels into three major levels of complexity.

In the first level, we have only B and I labels. Thus, every chunk has a beginning token labeled B and the rest of its tokens are labeled I. We do not need to use O label at this level as all the tokens in the sentence belong to at least one chunk.

In the second level, we used the specific types of chunks that each token belongs to. The set of labels are given in Table 1. So, for example, for a NP chunk, we label its initial token as B-NP and the rest as I-NP. We also used PUP to label punctuation marks.

In the third level, the labels of the second category are augmented with the semantic roles. For example, NP-SBJ marks a noun phrase which functions as the subject of a predicate. There are 16 such role labels.

In the treebank, one basic phrase label such as ADVP might cary several simultaneous semantic tags. We used only the first of those and discarded the rest. Also, we discarded the numeric tags identifying the relations among phrases. The set of augments are given in Table 2.

5 CRFs for Chunking

To model the statistical relations among the sentence tokens and the sequence of output labels, we used conditional random fields, (CRF), [18]. CRFs have proven to be powerful tools to model the conditional probability of output label sequence given a sequence of input word tokens in a sentence.

Let $x = (x_1, x_2, \ldots, x_n)$ be an input sequence of tokens in a Turkish sentence and $y = (y_1, y_2, \ldots, y_n)$ be a candidate sequence of output labels. For example, for the first level of labels explained in the previous section, we have $y_i \in \{B, I\}$.

The probability of Y given X is expressed as

$$P(y|x) = \frac{1}{Z(x)} \exp(\sum_{i=1}^{n} \sum_{j=1}^{m} \lambda_j f_j(y, x, i)),$$

Table 2. Semantic roles and functions

Role identifier	Description
-SBJ	Surface subject
-TMP	Temporal phrases
-LOC	Location
-DIR	Direction and trajectory
-PRD	Non VP predicates
-CLR	Closely related
-MNR	Manner
-TPC	Topicalized and fronted constituents
-EXT	Spatial extent of an activity
-NOM	Non NPs that function as NPs
-PUT	Marks the locativecomplement of put
-LGS	Logical subjects in passives
-TTL	Titles
-VOC	Vocatives
-DTV	Dative object
-PRP	Purpose and reason

where $f_j(y, x, i)$ is the j^{th} feature function corresponding to the token i in the sentence and $Z(x)$ is a normalization factor.

We used `wapiti` for the implementation of CRF, [19]. `wapiti` provides a fast training and labeling and uses the standard feature templates used by popular implementations like `CRF++`. The default options of `wapiti` uses l_1 regularization for fast convergence.

6 Chunking Features

An analysis at the syntactic level of a Turkish sentence needs to use the morphemes of the words as the morphological structure of Turkish words are closely related their syntactic functions in the sentence. For agglutinative languages like Turkish, morphological analysis is an essential early step in chunking as well as any other type of analysis. Turkish words may be quite long and contain a mixture of derivational and inflectional morphemes. In chunking, we use the inflectional morphemes and the word root.

There are a few available tools that perform automatic morpheme decomposition with a high level of accuracy, [20]. In our work, we used our own FST based morphological analyzer together with manual disambiguation. Thus, using the gold morphology, our chunking performance is isolated from the errors in morphological analysis.

In Turkish, cases indicate the syntactic function of a noun in the sentence. The case markings are suffixed to the heads of phrases. Thus, the presence of a case marking on a noun indicates that the phrase ends at that noun. Exception to this heuristic is the presence of genitive marking. In Turkish, genitive marking identifies the possessor in the noun compound. Thus, genitive is usually found when its noun is not the end of its phrase.

The basic features are the tokens themselves. This basic choice defines as many features as there are distinct word tokens. Contextual features for the token are inferred from the tokens around it.

We also tried the POS tags of the tokens as features and include contextual POS tags as well.

A summary of the features that we used in our training and test is given in Table 3.

Most of the features in the table are self explanatory. The feature F18 includes a binary feature A_i indicating whether the next root is an auxiliary Turkish verb. In Turkish, it is very common to construct two-word verbs by combining a noun or adjective with verbs "et" (do), "yap" (do) and "ol" (be). The variable A_i identifies the presence of one these auxiliary roots in the i^{th} word.

Table 3. Features

Identifier	Feature	Definition
F0	p_i	Current POS
F1	p_{i+1}	Next POS
F2	p_{i-1}	Previous POS
F3	c_i	Current case
F4	c_{i+1}	Next case
F5	c_{i-1}	Previous case
F6	g_i	Current has genitive marking, binary
F7	g_{i-1}	Previous has genitive marking, binary
F8	s_i	Current has possessive marking, binary
F9	s_{i+1}	Next has possessive marking, binary
F10	s_{i-1}	Previous has possessive marking, binary
F11	r_i	Current root
F12	r_{i+1}	Next root
F13	r_{i-1}	Previous root
F14	$U_i p_i p_{i-1}$	Current initial case, current and previous POS's
F15	$c_{i-1} s_i p_{i-1}$	Previous case, current has possessive, previous POS
F16	$c_i s_{i+1}$	Current case, next has possessiv
F17	$r_{i+1} p_{i+1}$	Next root , next POS
F18	$A_{i+1} p_{i-1}$	Next has auxiliary verb, previous POS

7 Experiments

For CRF based tagging, we use the features in Table 3. In listing the scores, P denotes the precision, R, recall and F, the F-measure.

For the first level of granularity, the possible output labels are B and I. The baseline is when we use only the tokens of the sentence as features. This choice of feature set gives the token level baseline accuracy of 0.68.

When we use the full set of features in Table 3, we obtain the results given in Table 4, broken down into performances for each label. The token level accuracy is 0.88.

Table 4. Results when only the boundaries of the chunks are identified

Tags	P	R	F
B	0.88	0.89	0.88
I	0.88	0.88	0.88

In the second level, we try to identify the type of each chunk. For this, we use the output labels such as B-NP, I-NP etc. Considering all of the 9 tags in Table 1, we have 17 possible output classes for the CRF tagger. As the number of classes increase, the small data size starts to become a problem in training. In order to mitigate the effects of data sparsity, we first analyzed our training data and kept only the most frequent labels, grouping the rest under a common label, O. The distribution of the labels in this new setup is given in Table 5. There are a total of 33,101 tokens in the training set. Note that the most common 5 labels make up the 94% of all the labels.

Table 5. Distribution of labels

Chunk label	**Percentage**
NP	35.9
VG	15.5
PUP	15.3
S	14.3
PP	12.4
ADVP	3.9
ADJP	1.5
CC	1.2

We drop the last three labels and classify them as O. The results under this new setup are given in Table 6. The token level accuracy is 0.66.

Table 6. Performance of the tagger broken down into chunk types.

Tags	P	R	F
B-NP	0.68	0.77	0.72
I-NP	0.56	0.78	0.65
B-VG	0.78	0.80	0.79
I-VG	0.64	0.38	0.47
PUP	0.96	0.99	0.98
B-S	0.43	0.21	0.28
I-S	0.49	0.38	0.43
B-PP	0.44	0.32	0.37
I-PP	0.56	0.45	0.50
O	0.62	0.58	0.60

As expected, the token level accuracy is drastically lower in this level.

Starts of the NPs are identified with fairly high accuracy. Inside the NP the performance somewhat declines. For VG, again the starts are handled better. This is somewhat expected as most verb groups in Turkish contain a single word. The low accuracy of S label is somewhat expected as S actually is a crude chunk whose boundaries are delineated by other chunks such as VG and NP.

To further analyze the source of errors, we constructed the confusion matrix for the output labels. The matrix is given in Table 7.

Table 7. Confusion matrix for the second level of granularity

	B-NP	I-NP	B-VG	I-VG	PUP	B-S	I-S	B-PP	I-PP	O
B-NP	0	**79**	8	3	3	32	22	27	1	14
I-NP	**32**	0	12	36	**11**	3	**86**	6	72	32
B-VG	**26**	**28**	0	15	1	1	24	14	2	17
I-VG	8	**95**	59	0	5	0	45	3	22	15
PUP	0	6	0	0	0	0	1	0	0	0
B-S	90	30	2	1	3	0	21	7	1	14
I-S	28	**279**	34	21	**15**	8	0	14	50	41
B-PP	76	32	4	0	1	9	26	0	13	15
I-PP	11	**184**	7	4	0	1	55	17	0	12
O	28	59	13	7	0	4	36	18	25	0

Looking at the confusion matrix, PUP column indicates that, in some error cases, other labels are predicted as punctuation when they are not. PUP is a label that is easy to learn by the CRF. However, the surrounding context seems to confuse the learner. This usually happens when a punctuation occurs within a noun phrase or subordinate sentence.

We also see that noun phrases are the source of many errors. In particular, S and NP are confused with each other. This is due the presence of NP's in the subordinate sentence boundaries. Similarly, NP frequently occurs within PP's which confuses the identification PP boundaries.

Another interesting source of errors is related to the verb group, VG. In many cases, Turkish verb group is a two word compound formed by a noun and verb such as "yardım ettim", "I helped". The first noun confuses the verb group with noun phrase. Similarly, VG-S confusion is a source of error. Again, NP's at the subordinate sentence boundaries confuse the VG identification.

In the last set of experiments, we used the augmented labels. Obviously, this increases the number of labels and as a result decreases the accuracy for each label. The sparsity becomes more prominent. As in the previous granularity level, we analyze the label counts in the training set and keep the most frequent labels as distinct classes and lump the rest together as O class. Instead of giving a large table of percentages, here we shortly describe the distribution. There are 53 distinct labels (not counting the B- and I- distinction). 17 of those make up 95% of all the labels. So we collect under O, less frequent 36 labels making up the remaining 5% of all the labels.

Table 8. Performance when the semantic roles and functions are identified

Tags	P	R	F
B-NP-SBJ	0.73	0.93	0.82
I-NP-SBJ	0.51	0.71	0.59
B-VG	0.70	0.82	0.76
I-VG	0.52	0.39	0.45
PUP	0.94	1.00	0.97
B-NP	0.20	0.17	0.18
I-NP	0.30	0.29	0.29
B-S	0.35	0.20	0.25
I-S	0.37	0.40	0.39
B-NP-PRD	0.11	0.05	0.07
I-NP-PRD	0.27	0.16	0.20
B-PP-CLR	0.24	0.13	0.17
I-PP-CLR	0.32	0.18	0.23
B-PP	0.47	0.13	0.20
I-PP	0.46	0.39	0.42
B-PP-DIR	0.68	0.61	0.64
I-PP-DIR	0.81	0.56	0.66
B-ADVP	0.55	0.59	0.57
I-ADVP	0.55	0.32	0.41
B-PP-LOC	0.28	0.23	0.25
I-PP-LOC	0.53	0.30	0.38
B-S-TPC	0.19	0.17	0.18
I-S-TPC	0.38	0.23	0.29
B-PP-TMP	0.61	0.25	0.35
I-PP-TMP	0.46	0.40	0.43
B-ADJP-PRD	0.08	0.03	0.05
I-ADJP-PRD	0.36	0.19	0.25
B-ADVP-TMP	0.68	0.45	0.54
I-ADVP-TMP	0.60	0.41	0.49
O	0.48	0.45	0.46

The results under this setup are given in Table 8. We have a token level accuracy of 0.59. We did not include the scores for labels that never occur in the test set.

Again, the highest performance is observed for the starts of subject noun phrases and verb groups. Interestingly, B-NP-SBJ is even higher than the accuracy of B-NP in the previous experiment. This might be due the bias created by the unmarked SOV order of Turkish sentences which places the subjects at the start of the sentences.

Again the most common source of errors is the confusion of NP's with other chunks. In particular, NP-SBJ is often confused with the generic class NP. Another interesting point to note is the difference between the performances of PP-DIR and PP. The learner detects PP-DIR better using the ablative and dative case markings.

The full confusion matrix for the third level of granularity is given in the supplementary materials of this paper.

8 Conclusion

In this work, we attempted a CRF based approach to Turkish chunking. Using the morpheme level features, we reported performances for different levels of chunk identification.

We used a novel approach to generate training and test data by leveraging the translated trees of the Penn Treebank. In previous works on Turkish chunking, dependency treebank was used [14]. Although we used a moderate sized data set, our approach in data generation is general enough to increase the size of the chunking corpus as more translation data becomes available.

Compared to previous works on Turkish chunking, our present work is the first attempt to solve the general chunking problem. In [12], only the NP chunks are detected using hand-crafted rules. [13] considers verb chunks only. All the other tokens in a sentence are considered out of chunk. Moreover, [13] uses dependency labels of word tokens as a feature. This requires an accurate dependency parser as a preliminary stage of the chunking. In our work, no such assumptions are made. Of course, such a general approach reflects negatively on the performance scores.

Many of our features in Table 3 have previously been used in similar chunking tasks for other languages the literature. However, we believe their application to Turkish is novel. In particular, our use of genitive and possessive markings as features is new.

An obvious direction for improving the results is designing better features to reflect the syntactic dependencies of the words in a Turkish chunk. Rather than using the given POS and case tags, one can define custom categories to reflect the coherence of nearby words. We actually used such features using the presence or absence of genitive and possessive markings. Another approach would be to design features to discriminate most frequently confused elements such as NP-PP and VG-PP. Such features would have to reflect the peculiar structure of Turkish syntactic constituents.

References

1. Abney, S.: Parsing by chunks. In: Principle-Based Parsing, pp. 257–278. Kluwer Academic Publishers (1991)
2. Jurafsky, D., Martin, J.H.: Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition. Prentice Hall Series in Artificial Intelligence, 2 edn. Prentice Hall (2009)
3. Ramshaw, L.A., Marcus, M.P.: Text chunking using transformation-based learning. In: Third ACL Workshop on Very Large Corpora, pp. 82–94 (1995)
4. Kudo, T., Matsumoto, Y.: Chunking with support vector machines. In: Proceedings of the Second Meeting of the North American Chapter of the Association for Computational Linguistics on Language Technologies, NAACL 2001, pp. 1–8. Association for Computational Linguistics, Stroudsburg (2001)
5. Zhang, T., Damerau, F., Johnson, D.: Text chunking based on a generalization of winnow. J. Mach. Learn. Res. 2, 615–637 (2002)
6. Tjong Kim Sang, E.F., Buchholz, S.: Introduction to the CoNLL-2000 shared task: Chunking. In: Proceedings of the 2Nd Workshop on Learning Language in Logic and the 4th Conference on Computational Natural Language Learning, ConLL 2000, pp. 127–132. Association for Computational Linguistics, Stroudsburg (2000)

7. Park, S.B., Zhang, B.T.: Text chunking by combining hand-crafted rules and memory-based learning. In: Proceedings of the 41st Annual Meeting on Association for Computational Linguistics, ACL 2003, vol. 1, pp. 497–504. Association for Computational Linguistics, Stroudsburg (2003)
8. Lee, Y.-H., Kim, M.-Y., Lee, J.-H.: Chunking using conditional random fields in korean texts. In: Dale, R., Wong, K.-F., Su, J., Kwong, O.Y. (eds.) IJCNLP 2005. LNCS (LNAI), vol. 3651, pp. 155–164. Springer, Heidelberg (2005)
9. Gune, H., Bapat, M., Khapra, M.M., Bhattacharyya, P.: Verbs are where all the action lies: Experiences of shallow parsing of a morphologically rich language. In: Proceedings of the 23rd International Conference on Computational Linguistics: Posters, COLING 2010, pp. 347–355. Association for Computational Linguistics, Stroudsburg (2010)
10. Chen, W., Zhang, Y., Isahara, H.: An empirical study of chinese chunking. In: Proceedings of the COLING/ACL on Main Conference Poster Sessions, COLING-ACL 2006, pp. 97–104. Association for Computational Linguistics, Stroudsburg (2006)
11. Sun, G.L., Huang, C.N., Wang, X.L., Xu, Z.M.: Chinese chunking based on maximum entropy markov models. International Journal of Computational Linguistics & Chinese Language Processing 11, 115–136 (2006)
12. Kutlu, M.: Noun phrase chunker for Turkish using dependency parser. Master's thesis, Sabancı University (2010)
13. El-Kahlout, İ.D., Akın, A.A.: Turkish constituent chunking with morphological and contextual features. In: Gelbukh, A. (ed.) CICLing 2013, Part I. LNCS, vol. 7816, pp. 270–281. Springer, Heidelberg (2013)
14. Atalay, N.B., Oflazer, K., Say, B.: The annotation process in the Turkish treebank. In: 4th International Workshop on Linguistically Interpreted Corpora (2003)
15. Marcus, M.P., Marcinkiewicz, M.A., Santorini, B.: Building a large annotated corpus of English: The Penn treebank. Computational Linguistics 19, 313–330 (1993)
16. Kornfilt, J.: Turkish. Routledge (1997)
17. Yıldız, O.T., Solak, E., Görgün, O., Ehsani, R.: Constructing a Turkish-English parallel treebank. In: Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pp. 112–117. Association for Computational Linguistics, Baltimore (2014)
18. Sha, F., Pereira, F.: Shallow parsing with conditional random fields. In: Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology, NAACL 2003, pp. 134–141. Association for Computational Linguistics, Stroudsburg (2003)
19. Lavergne, T., Cappé, O., Yvon, F.: Practical very large scale CRFs. In: Proceedings the 48th Annual Meeting of the Association for Computational Linguistics (ACL), pp. 504–513. Association for Computational Linguistics (2010)
20. Hakkani-Tur, D., Oflazer, K., Tür, G.: Statistical morphological disambiguation for agglutinative languages. Computers and the Humanities (2002)