Shallow Parsing in Turkish

Türkçe'de Sığ Çözümleme

Ozan Topsakal[1], Onur Açıkgöz[1], Ali Tunca Gürkan[1], Ali Buğra Kanburoğlu[1]
Burak Ertopçu[1], Berke Özenç[1], İlker Çam[1]
Begüm Avar[2], Gökhan Ercan[1], Olcay Taner Yıldız[1]
ozan.topsakal@isik.edu.tr, onur.acikgoz@isik.edu.tr, ali.gurkan@isik.edu.tr, bugra.kanburoglu@isik.edu.tr
burak.ertopcu@isik.edu.tr, berke.ozenc@isik.edu.tr, ilker.cam@isik.edu.tr
begum.avar@boun.edu.tr, gokhan.ercan@isik.edu.tr, olcaytaner@isikun.edu.tr
[1]Department of Computer Engineering, Işık University, İstanbul, Turkey
[2]Department of Linguistics, Boğaziçi University, İstanbul, Turkey

Abstract—In this study, shallow parsing is applied on Turkish sentences. These sentences are used to train and test the per-formances of various learning algorithms with various features specified for shallow parsing in Turkish.

Kewords—Natural Language Processing; NLP, Shallow Parsing; Turkish

Özetçe —Bu çalışmada, Türkçe cümleler üzerinde sığ çözümleme yapılmıştır. Bu cümleler, Türkçe sığ çözümleme için belirlenmiş çeşitli özelliklerle çeşitli öğrenme algoritmalarının eğitilmesi ve performanslarının test edilmesi için kullanılmıştır.

Anahtar Sözcükler—Doğal Dil İşleme; DDİ; Sığ Çözümleme; Türkçe

I. INTRODUCTION

All of the living species in nature communicate with each other. Human beings communicate with words, and when they do, lots of neural input is calculated in the brain. From the 1950's scientists have been trying to imitate this this neural linguistic network on computers, which paved the way for the emergence of Natural Language Processing (NLP) as a branch of computer science. At the heart of NLP lies the goal to make computers understand human verbal communication and thus to improve human-computer interaction. Three important fields of computer science, namely computational linguistics, machine learning, and programming, are gathered together to achieve this goal. Hence, today, NLP is one of the most attractive fields for computer scientists around the globe.

Most of the NLP studies focus on analytic languages like English and many other Indo-European languages, and studies on agglutinating languages like Turkish are limited in this field. Agglutinative languages, in general, are arguably more difficult to work on than others, due to the fact that a word may get numerous different meanings via the use of morphological markers, such as suffixes.

In general there are two approaches in NLP. One of them is semantic analysis and the other one is syntactic analysis, which shallow parse is an extension of. In order to understand the syntactic structure of a sentence, a computer has to understand not only the meanings of individual words, but also their hierarchical sequence in the sentence. In many studies on NLP, the most difficult challenge is parsing. Some of the tools are slow and give too much information, and some others are fast but give insufficient information about the sentence. It is, therefore, a burden for NLP studies to find the balance between those two. Shallow Parse (partial parse) can provide the necessary information in a reasonable time.

In this study, our aim is to create several models using different features and classifiers for the Shallow Parse task. Next, by testing those and comparing the results, we will try to find the model that generates the best outcome.

This paper is organized as follows: We define Shallow Parsing problem in Section II and give the previous work in Section III. In continuous models, we represent words with continuous vectors, namely word embeddings. A brief introduction to word embeddings is given in Section IV. The details of our dataset and how it is constructed are given in Section V. We give our experiment methodology in Section VI and results in Section VII. Lastly, we conclude in Section VIII.

II. SHALLOW PARSING

Many language processing tasks do not require complex parse trees. Instead, a partial parse, or a shallow parse of a sentence is sufficient. Shallow parsing is the process of identifying flat non-overlapping parts of a sentence. These parts typically include Özne, Yüklem, Nesne, Zarf Tümleci, and Dolaylı Tümleç. Since a parsed text does not include a hierarchical structure, a bracketing notation is sufficient to denote the location and the type of shallow parse chunks in a sentence. Here is a sample text with shallow parse chunks marked:

[OZNE Türk Hava Yolları] [ZARF T UMLECI Salı günü] yeni [NESNE indirimli fiyatlarını] [Y UKLEM açıkladı]

This sentence contains 5 shallow parse chunks including 3 words labeled as ÖZNE (SUBJECT), 2 words labeled as ZARF TÜMLECİ (ADVERBIAL CLAUSE), 3 words labeled as NESNE (DIRECT OBJECT), and 1 word labeled as YÜKLEM (PREDICATE). Table I shows typical shallow parse tags and the questions asked to the predicate to identify the chunks for those tags.

978-1-5386-0930-9/17/$31.00 ©2017 IEEE

Table I. LIST OF SHALLOW PARSE CHUNK TAGS

Tag	Question	Example
ÖZNE	Who	**Atatürk** yurdu düşmanlardan kurtardı.
ZARF TÜMLECİ	When	IMKB **Salı günü** 60 puan yükseldi.
DOLAYLI TÜMLEÇ	Where	Bugün **evde** kalmayacağım.
NESNE	What	Bu hafta **tatil** yapacağım.
YÜKLEM	Predicate	Geçen gün 3000 TL **kazandık**.

In shallow parsing, one tries to find the strings of text that belong to a chunk and to classify the type of that chunk. Standard approach for shallow parsing is a word-by-word classification, where the classifier is trained to label the words in the text with tags that indicate the presence of particular chunks. After giving the class labels to our training data chunk labels, the next step is to select a group of features to discriminate different chunks for each input word. Table II shows the sample text represented with chunk labels and three possible features, namely the root form of the word, the part of speech (POS) tag of the word, and a boolean feature for checking the capital case.

Table II. SHALLOW PARSING AS A CLASSIFICATION PROBLEM

Word	Features				Label
	Root	Pos	Capital	...	
Türk	Türk	Noun	True	...	ÖZNE
Hava	Hava	Noun	True	...	ÖZNE
Yolları	yol	Noun	True	...	ÖZNE
Salı	Salı	Noun	True	...	ZARF TÜMLECİ
günü	gün	Noun	False	...	ZARF TÜMLECİ
yeni	yeni	Adjective	False	...	NESNE
indirimli	indirimli	Adjective	False	...	NESNE
fiyatlarını	fiyat	Noun	False	...	NESNE
açıkladı	açıkla	Verb	False	...	YÜKLEM
.	.	Punctuation	False	...	HİÇBİRİ

Given such training data, a classifier like a neural network or a decision tree can be trained to label new sentences. Figure 1 shows the operation of such a classifier at the point where the word *günü* is next to be labeled. For this classifier, the window size is 2, that is, we assume a context window that includes two preceding words and two succeeding words.

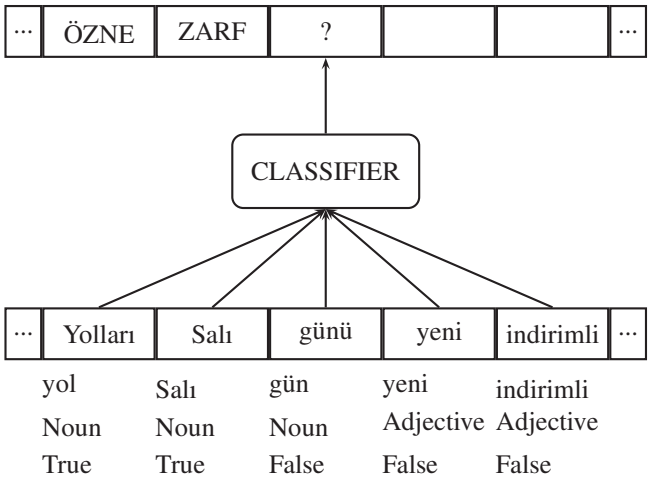

Figure 1. Classifier based approach to shallow parsing. The parser slides a context window over the sentence, classifying words as it proceeds. At this point, the classifier is attempting to label *günü*. Features derived from the context typically include the words, POS tags, etc.

III. PREVIOUS WORK

A. Linguistic Background

In linguistics, sentences are assumed to be constructed from a rule-based combination of several constituents. In other words, sentences are not considered as linear strings of words - rather, they are treated as hierarchically organized structures consisting of phrases (such as NPs, i.e. noun phrases, and VPs, i.e. verb phrases) which may, in turn, contain other phrases. Determining the constituents of a sentence is a renowned problem in syntax. Various constituency tests have been proposed in the linguistics literature, yet they are not always reliable or may lack a cross-linguistic applicability [3].

At the very basics, a sentence consists of a subject NP and a predicate VP. The predicate, in turn, may contain a variety of phrases with several grammatical roles, such as the direct or indirect object, or an oblique object. Turkish is typically treated as a head-final subject-object-verb (SOV) language, yet a central problem with Turkish is that it does not have a strict word order. In other words, the constituents of a sentence can be scrambled such that they can occur in any order with slight modifications in the sentential meaning. A particular order is chosen mainly on pragmatic grounds [8].

In general, language in its written form gives us orthographic clues about word boundaries but not about the phrase or constituent boundaries (except for punctuation marks in some cases). In speech, prosodic cues such as stress and intonation patterns may help the listener in parsing, yet such cues are unavailable in written texts. In Turkish, the problem of parsing is even more advanced, as not only content words in their base forms but also functional morphemes, such as affixes, may be the source of ambiguity which cannot be resolved without the context. One of the central foci of literature on Turkish linguistics has been the complexity of Turkish morphology. Turkish is a textbook example for an agglutinative language, i.e. words in their surface form may contain various morphemes, especially suffixes. While certain affixes have clear functions/meanings, there exist others for which the meaning can only be determined within a discourse. Derivational affixes may change the POS of a word, which makes it possible for a root to serve grammatical roles that are not typical for the its root's POS-tag. Even without any derivational process, the POS distinction among the nominal words (especially between nouns and adjectives) is far from being absolute. For instance an adjective can be used as a noun and hence can occupy the subject position (the head of NP) within a sentence.

Zengin fakir-in hal-in-i anla-maz.

rich poor-Gen state-Poss-Acc understand-Neg

Relevant to Shallow Parsing tasks are especially the problems with inflection. Typically, inflectional affixes have well-defined and transparent grammatical functions such as marking the number or person of the base they are attached to. Structural case markers are especially useful in detecting the grammatical role a constituent plays in a sentence: Nominative (*Ø*) marks the subject (i.e. is attached to the word which is the head of the subject NP), whereas Accusative (*-(y)I*) marks the direct object of a sentence. Yet there are two central problems with their reliability. First, there exist also lexical

(or inherent) case markers which are selected by the verb and are not syntactically predictable [4]. For instance, a verb like *döv-* 'to beat' gets an accusative-marked direct object (structural case) while *vur-* 'to hit' requires a dative-marked direct object (lexical case). Dative, in other cases, is mostly used to mark the indirect object (for ditransitive verbs) or an oblique object denoting orientation of the action/event. The second problem relates to the fact that the accusative is not always overtly marked in Turkish. The choice between the zero- vs. accusative-marked direct object is typically assumed to depend on the definiteness, referentiality and/or specificity of the object-NP [12], [9], [10], [5].

Another issue in Turkish syntax that leads to further complications for Shallow Parsing is related to the fact that Turkish is a pro-drop language, i.e. pronouns may be omitted when they are predictable from the pragmatical and/or grammatical context. This property of Turkish leads to cases where the antecedent of an agreement relationship (in terms of person and number) is missing on the surface and can only be retrieved through context. Related to this linguistic property, Turkish sentences can have null subjects, i.e. the subject is not realized yet still consists of a hidden NP.

A final burden for NLP studies in Turkish relates to embedding, phrases may be embedded in other phrases, sentences (or clauses) in other sentences. Hence, a sentence - in theory - may be infinitely long. In Turkish, there are three types of embedded sentences: (i) Those that are syntactically marked, which are similar to embeddings in Indo-European languages like English *that*-clauses, (ii) those that are morphologically marked with a nominalizer attached to the verb of the embedded sentence, and (iii) those that are unmarked on the surface [8]. The verb in type (ii) embedded sentences is in non-finite form which may lead to complications in annotation and parsing. Moreover, the free word-order of Turkish makes it difficult to predict which constituent of which sentence occurs in what position.

In short, Turkish, with its complex morphology and unstable word order, provides various challenges for NLP studies in general, and Shallow Parsing in particular.

B. Computational Background

In their paper, Eichler and Neumann [6] propose a system for extracting noun groups among various constituents of sentences. They use a sophisticated parser with several constraints. They do not, however, analyze Turkish data.

In another study, Yildiz et. al. [16] manually extract data from Penn Treebank. Upon translating the data into Turkish, they try to automatically identify and tag the chunks. For training data, they use conditional random fields (CRF). Aiming to solve the general chunking problem, they report different levels of chunk resolution.

Kutlu & Cicekli [11] work on noun phrase chunking in Turkish. They use hand-crafted rules for the dependency parser that are suitable for complex sentences due to their flexibility. In this way, they arguably get better results for shallow parsing of all phrase types.

Finally, El-Kahlout & Akin [7] argue that while chunking is relatively unproblematic for simplex words and for analytic languages like English, Turkish is a morphologically rich language and therefore NLP tasks in Turkish require more elaborated features. Two different techniques are proposed for Turkish. In the former, chunks are extracted according to the results of the Turkish dependency parser. In the latter, annotated Turkish sentences are used by a CRF-based chunker which is enhanced with morphological and combinatorial features. According to the results of various experiments, CRF-based chunking with additional properties is reported to perform better (with an F-measure of 87.5 in general, and up to 91.95 for verb chunks) than baseline chunker with basic features or dependency-based chunker.

IV. WORD EMBEDDINGS

Mikolov et al.'s SkipGram is an unsupervised neural network based distributional semantic model (DSM). Main intuition is to learn distributed word representations (i.e., word embeddings) through maximizing the probability of surrounding words within a window by learning weights of each word vector in context dimensions. SkipGram converts supervised learning problem into a unsupervised learning one by turning positive and negative neighbors of words to probabilities under the Markov assumption which is the basic idea behind the traditional discrete language models.

Continuous bag-of-word (CBOW) model is also proposed by Mikolov et al. which is reported to be more scalable than SkipGram model with some little efficiency loss on semantic tasks. While SkipGram predicts surrounding words of a current word w, CBOW model predicts the w based on the context. Due to the fact that CBOW model ignores word orders, Mikolov et al. [13] reported that it performs worse on semantic tasks.

V. DATA

Our dataset originates from Penn Treebank corpus. It is the first corpus populated with American English texts. Originally it contains about 1 million words in 15 categories and has received much attention in computational linguistics field. In this study, we used 1400 sentences from the Penn Treebank corpus (containing 13194 words, including punctuations) which are carefully translated into Turkish [17].

A. Morphological Disambiguation

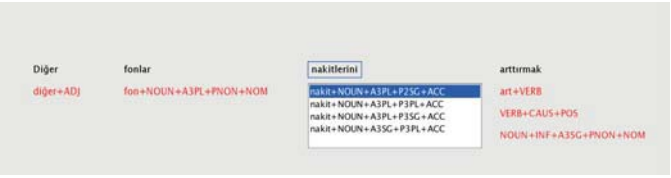

Figure 2. Morphological disambiguation tool

Turkish is an agglutinative language in which words are formed by attaching derivational and inflectional suffixes to the roots. Morphemes added to a word can change its part of speech, i.e., for instance, convert a noun to a verb - or vice-versa - or can create adverbs from adjectives. Moreover, during word formation, some letters can change or undergo deletion. Hence, without determining the lemma of a word from its

surface form, based on its intended meaning, it is not possible to identify the word correctly and to extract candidate senses from a dictionary.

Figure 3. Annotation Tool

Following the translation, the corpus has been morphologically disambiguated. In that work, human annotators selected the correct morphological parsing among multiple possible analyses returned from the automatic parser (See Figure 2 for the morphological disambiguation tool used). The tag set and morphological representation were adopted from the study [14]. Each output of the parser comprises the root of the word, its part-of-speech tag and a set of morphemes, each separated with a "+" sign.

B. Shallow Parse Tagging

Table III. TAGS AND THEIR OCCURRENCES

Tag	Occurrence (word)
HİÇBIRI	3445
NESNE	2710
ÖZNE	2177
YÜKLEM	2037
ZARF TÜMLECI	1879
DOLAYLI TÜMLEÇ	935

Figure 3 illustrates the annotation tool we used. There are 6 different tags in this dataset, whose implementation is shown in Table III. Annotated dataset and source codes are freely accessible[1].

VI. METHODOLOGY

The six known classification algorithms we use, are:

- Dummy: As a baseline classifier, we use the Dummy classifier, which decides based on the prior class probability without looking at the input. All test instances are assigned to the class with the maximum prior. It is not a learning algorithm in the usual sense, and *any* plausible learning algorithm must have smaller error rates than Dummy; it is indeed surprising that Dummy is sometimes quite accurate.
- C45: The archetypal decision tree method [15].
- Knn: K-Nearest Neighbor classification algorithm that uses the Euclidean distance.
- Lp: Linear perceptron with softmax outputs trained by gradient-descent to minimize cross-entropy [2].
- Mlp: Well-known multilayer perceptron classification algorithm [2].
- Nb: Classic Naive Bayes classifier where each feature is assumed to be Gaussian distributed [1] and each feature is independent from other features.

[1] http://haydut.isikun.edu.tr/nlptoolkit.html

A. Discrete Model

This section covers the details about the features that are used in this study, together with the seven different models that we had built.

1) Features: In this section, all features, regardless of the feature sets, will be explained in detail.

- Case Attribute (C): This feature has a discrete value. It checks the word's case suffix (such as accusative, dative etc.) and, by default, if there is no case suffix, its value is "NULL". This is a strong feature because the case of a word can give ideas about its class label. In Turkish, for instance, direct object NPs (i.e. "nesne") are typically marked with accusative, whereas "dolaylı tümleç" usually are marked with dative, locative or ablative case.
- IsAdjective Attribute (IAD): This is a binary valued feature, that shows whether a word used as an adjective or not.
- IsAuxillaryVerb Attribute (IAU): This is a binary valued feature, that checks whether the word is an auxiliary verb. This is another strong feature because it may be useful in identifying the word as (part of) a predicate ("yüklem").
- Is Capital Letter (IC): This a binary valued feature. Its value returns "true" if the word starts with a capital letter, and "false" otherwise. Words with an initial capital letter are proper nouns unless the word is the first word of a sentence. Being a proper noun can be important for Shallow Parse labeling because, for instance, if it is a name of a person, it is more probable for the word to be the subject or the object, rather than the adverbial clause or the predicate, of the sentence it occurs in. Yet such clues are not always reliable.
- IsIndirectObjectMoneyAttribute (IIOM): This is a binary valued feature for a given word. If the word equals to "artarak" or "azalarak" and the named entity type of the word is "NONE", it returns true, otherwise false.
- LastMorphTagAttribute (LM): This is a discrete attribute which returns the morphological tag of the last inflectional group.
- LastSyllable Attribute (LS): This is a discrete valued feature, that finds the last 3 characters of the word, and it can be helpful in detecting suffixes.
- LastIGContainsTagAblative Attribute (LTAB): This is a binary valued feature, that checks whether the word is in ablative case or not.
- LastIGContainsTagAccusative Attribute (LTAC): This is a binary valued feature, that checks whether the word is in accusative case or not.
- LastIGContainsTagGenitive Attribute (LTG): This is a binary valued feature, that checks whether the word is in genitive case or not.
- LastIGContainsTagInstrumental Attribute (LTI): This is a binary valued feature, that checks whether the word is in instrumental case or not.

- LastIGContainsTagPossesive Attribute (LTP): This is a binary valued feature, that checks whether the word has a possessive marker or not.
- Main POS (MP): This is a discrete valued feature. The value of the feature consists of the POS (part-of-speech) tag of the word's root. POS tag represents the grammatical category of the word such as Verb, Noun, Adjective etc. This feature can be helpful in determining a word's Shallow Parse label, because, for example, if a word is a Verb, it has a strong possibility to be the predicate of the sentence. Yet such clues are not always reliable.
- NerTag Attribute (NT): This is a discrete valued feature, that finds the NER (Named Entity Recognition) tag of the word. This is another strong feature, especially for detecting subjects and objects. For Instance, if the word has a "PERSON" NER tag it is generally the subject or the object of the sentence.
- Root Form (RF): This is also another discrete valued feature and stores the root of the word as its value. A root is a word without any suffixes.
- RootPosAttribute (RP): This is a discrete attribute for a given word. It returns the POS tag of the word's root.
- Surface Form (SF): This is a discrete valued feature and stores the word itself as its value. It is a strong feature because the meaning of a word's root can be modified through affixation. In a word's surface form, the word is considered as a whole with all the affixes attached.

2) Methods: In this study, we used several machine learning algorithms to build our models. We have 6 different models and each of these have their own features. We give the details of the methods, such as the attributes, the classifier and its parameters in Table IV.

B. Continuous Model

We trained continuous word features from 1 million (M) word news corpus by using Mikolov et al.'s SkipGram and CBOW models. In order to measure effect of the corpus size to the model performances. Below is the hyper-parameters of the SkipGram and CBOW models we used:

- Context window (win): Word context window size where co-occurrence information is gathered. Default is 5.
- Dimension (d): Dimension size of the word embeddings. Number of neurons in neural network layers. Default is 100.
- Deleting infrequent words (del): Threshold frequency value for excluding words from the training. Default is 0.

We used two kind of morphological word forms:

- Surface Form (SU): Natural form of a word which appeared in a text as it is. Ex: *Zamanın çok ilerisinde bir bulu¸s?*
- Root Form (RO): Lexical root of a word gathered by morphological disambiguation in the sentence level. Ex: *Zaman çok ileri bir bul.*

VII. EXPERIMENTS

A. Inter-annotator Agreement

For evaluation of the annotated dataset, we used inter-annotator-agreement measure. Two different group of annotators annotated same sentences. Due to a lack of time, we could only re-annotate (by a different annotator) 100 of the total of 1400 sentences and got %79.0 inter-annotator agreement. As a comparison, the expected inter-annotator agreement, assuming the annotators annotated completely randomly, is %16.7.

B. Discrete Model

In the experiment phase, we adopted our dataset in Turkish language. Therefore, we had the advantage of a stratified 10-fold cross validation for our data.

We use the dataset of NLP toolkit of Isik University which contains 1400 annotated sentences, containing a total number of 13194 words. The annotation was done manually by several annotators. Some annotators were experts on grammar of Turkish while others are native speakers of Turkish with no special experience in this area. Therefore, it is possible that they annotated some words wrongly in complicated sentences.

Error rates for each method are shown in Table V.

C. Continuous Model

We set continuous models baseline as SkipGram (SG) model with hyper-parameter values win=5, d=100, del=0 (SG-win5-d100-del0). We trained 10 models in various configurations: 6 models for each corpus size (100K, 500K, 1M) in 2 different word forms (Surface, Root), a CBOW model, a model with higher context size 300 (1M-RO-d300), a model with lower context size 20 (1M-RO-d20), and a model with window size of 2. We also displayed discrete featured classifier results for benchmarking.

We ran our models with Dummy, Lp with learning rate: 0.1, Mlp with learning rate 0.1, hidden nodes = 50, Knn with k = 3, and Nb classifiers with 10-fold cross-validation on data. We ran all classifiers with the window size (w) of 1 except one (1M-RO-w2). Table VI shows classifier error rates and OOV percentages in our experiments:

According to our continuous model experiments:

- Word embeddings can provide valuable information as features of words for shallow parsing classification tasks. Although continuous features did not perform well standalone as manually handcrafted discrete features, they can potentially increase performance of existing models when used along with other discrete features.
- All continuous models performed worse on all classifiers except Knn compared to discrete single attribute baselines.
- Root form (RO) model performances do not depend on corpus size of the trained word embeddings. There

Table IV. DETAILS OF METHODS USED IN DISCRETE MODEL

Method	Attributes	Window	Classifier	Parameters	Author
METHOD1	C, IAD, IAU, LS, LTAC, LTG, LTI, LTP, MP, NT, RF, SF	2	Knn	k = 2	B.Ö.
METHOD2	C, IAD, IAU, IC, LS, LTAC, LTG, LTI, LTP, MP, NT, RF, SF	1	Nb		B.E.
METHOD3	C, IAD, IAU, IC, LM, LS, LTAB, LTAC, LTP, MP, NT, RF, RP, SF	1	Lp	learning rate = 0.1	A.B.K.
METHOD4	IC, IIOM, LTP, MP, RF	5	Mlp	learning rate = 0.1, number of hidden nodes = 50	A.T.G.
METHOD5	C, IAD, IAU, IC, LT, LTP, MP, RF, SF	1	Mlp	learning rate = 0.1, number of hidden nodes = 5, 8	O.T.
METHOD6	C, IAD, IAU, IC, IDT, LT, LTP, MP, RF, SF	1	Lp	learning rate = 0.1, number of hidden nodes = 10	O.A.

Table V. ERROR RATES OF METHODS USING STRATIFIED 10-FOLD CROSS-VALIDATION

Classifier	DUMMY	METHOD1	METHOD2	METHOD3	METHOD4	METHOD5	METHOD6
Error Rate	73.86	48.64	42.94	40.26	39.13	38.74	39.05

Table VI. ERROR RATES OF DIFFERENT METHODS USING 10-FOLD STRATIFIED CROSS-VALIDATION OF CONTINUOUS MODEL EXPERIMENTS

	OOV%	Dummy	Lp	Mlp	Nb	Knn
SURFACE (SU)						
Discrete Baseline	-	73.85	41.61	42.9	47.56	49.03
100K	32.4	73.6	46.74	45.65	55.31	50.67
500K	23.05	74.9	45.98	43.45	57.65	49.51
1M	20.31	75.11	46.87	44.58	58.02	50.68
ROOT (RO)						
Discrete Baseline	-	73.85	41.81	42.91	45.56	50.31
100K	20.77	75.23	46.77	43.53	56.44	49.32
500K	17.49	75.36	46.24	43.42	57.94	50.28
1M	16.33	75.38	46.99	44	57.94	50.92
OTHER CONF.						
1M-RO-CBOW	16.33	75.38	52.94	45.72	57.72	51.28
1M-RO-d300	16.33	75.38	47.16	43.47	68.28	52.89
1M-RO-d20	16.33	75.38	49.05	45.71	55.37	50.92
1M-RO-w2	25.45	75.12	46.84	43.06	61.63	51,78
MIN	16.33	73.6	41.61	42.9	45.56	49.03

is no significant effect of the corpus size to model performances on surface form models.

- Vector dimensions (d) of word embeddings did not affect shallow parsing performances significantly.
- Increasing window size did not increase model per-formances.

VIII. CONCLUSION

Shallow Parsing is an analysis of a sentence that identifies the constituents which are subject, object and some other word groups with particular grammatical functions. Shallow parsing was applied to 1400 Turkish sentences by 7 annotators. Performances of 6 different models using classifiers (which are Decision Tree (C45), Naive Bayes, Knn, Linear Perceptron, Multilayer Perceptron) for shallow parsing were observed.

We also used Mikolov et al.'s SkipGram and CBOW models for learning continuous representation of words. Then we used our trained word embeddings to feed our classifiers as continuous features of words without doing any feature enhancement. Our results show that, these continuous models can potentially increase shallow parsing classification task performance when used along with discrete features. We leave this for future work.

ACKNOWLEDGEMENTS

This work was supported by I¸sık University BAP projects 14B206 and 15B201. All authors contributed equally to this work. O. A., İ. Ç., B. E., A. T. G., A. B. K., B. Ö., O. T. designed and implemented discrete model experiments. They also labeled the data and wrote the manuscript. G. E. designed and implemented continuous model experiments. B. A. wrote the manuscript. O. T. Y. supervised the project, gave conceptual advice and wrote the manuscript.

REFERENCES

[1] E. Alpaydın, *Introduction to Machine Learning*, 3rd ed. The MIT Press, 2010.

[2] C. M. Bishop, *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.

[3] A. Carnie, *Syntax: A Generative Introduction*. Blackwell, 2006.

[4] N. Chomsky, *Lectures on Government and Binding*. Dordrecht: Foris, 1981.

[5] M. Dede, "Definiteness and referentiality in Turkish verbal sentences," in *Studies in Turkish linguistics*. Amsterdam: Benjamins, 1986, pp. 147–164.

[6] K. Eichler and G. Neumann, "Bootstrapping noun groups using closed-class elements only," in *LWA2010 - Workshop-Woche: Lernen, Wissen Adaptivitaet*, 2010.

[7] I. D. El-Kahlout and A. A. Akin, "Turkish constituent chunking with morphological and contextual features," in *CICLing*, 2013, pp. 270–281.

[8] E. T. Erguvanlı, *The function of word order in Turkish Grammar*. Berkeley: University of California Press, 1984.

[9] J. Kornfilt, "Case marking, agreement, and empty categories in Turkish," Ph.D. dissertation, Harvard University, 1984.

[10] ——, *Turkish*. London: Routledge, 1997.

[11] M. Kutlu and I. Cicekli, "Noun phrase chunking for Turkish using a dependency parser," in *ISCIS*, 2015, pp. 381–391.

[12] G. Lewis, *Turkish Grammar*. Oxford: Clarendon, 1967.

[13] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[14] K. Oflazer, B. Say, D. Hakkani-Tür, and G. Tür, "Building a Turkish treebank," in *Building and exploiting syntactically-annotated corpora*. Dordrecht: Kluwer, 2003.

[15] J. R. Quinlan, *C4.5: Programs for Machine Learning*. San Meteo, CA: Morgan Kaufmann, 1993.

[16] O. T. Yildiz, E. Solak, R. Ehsani, and O. Görgün, "Chunking in Turkish with conditional random fields," in *CICLing*, 2015, pp. 173–184.

[17] O. T. Yildiz, E. Solak, O. Gorgun, and R. Ehsani, "Constructing a Turkish-English parallel treebank," in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*. Baltimore, Maryland: Association for Computational Linguistics, June 2014, pp. 112–117.