An Open, Extendible, and Fast Turkish Morphological Analyzer

Olcay Taner Yıldız[1], Begüm Avar[2], Gökhan Ercan[1]
[1] Department of Computer Engineering, Işık University, İstanbul, Turkey
[2] Department of Linguistics, Boğaziçi University, İstanbul, Turkey
olcaytaner@isikun.edu.tr, begum.avar@boun.edu.tr
gokhan.ercan@isik.edu.tr

Abstract

In this paper, we present a two-level morphological analyzer for Turkish which consists of five main components: finite state transducer, rule engine for suffixation, lexicon, trie data structure, and LRU cache. We use Java language to implement finite state machine logic and rule engine, Xml language to describe the finite state transducer rules of the Turkish language, which makes the morphological analyzer both easily extendible and easily applicable to other languages. Empowered with a comprehensive lexicon of 54,000 bare-forms including 19,000 proper nouns, our morphological analyzer is amongst the most reliable analyzers produced so far. The analyzer is compared with Turkish morphological analyzers in the literature. By using LRU cache and a trie data structure, the system can analyze 100,000 words per second, which enables users to analyze huge corpora in a few hours.

1 Introduction

Morphological analysis is one of the key components of computational language processing, especially in morphologically rich languages. After some preprocessing stages such as stripping nontextual parts from the corpus and sentence segmentation, the first and foremost stage in computational analysis of the text is morphological analysis, that is dividing the words into constituent morphemes.

In this study, we deal with the morphology of Turkish, which is a textbook example for an agglutinative language. In Turkish, a word in its surface form contains 3 to 4 morphemes on the average (Sak, 2011), and these morphemes can have a semantic and/or syntactic content. Not only a surface form can have a multimorpheme structure in Turkish, but also has multi morphological analyses.

In this paper, we will present a new morphological analyzer, which is (i) open: The latest version of source codes, the lexicon, and the morphotactic rule engine are all available on the Internet[1], (ii) extendible: One of the disadvantages of other morphological analyzers is that their lexicons are fixed or unmodifiable, which prevents to add new bare-forms to the morphological analyzer. In our morphological analyzer, the lexicon is in text form and is easily modifiable, (iii) fast: Morphological analysis is one of the core components of any NLP process. It must be very fast to handle huge corpora. Compared to other morphological analyzers, our analyzer is capable of analyzing hundreds of thousands words per second, which makes it one of the fastest Turkish morphological analyzers available.

2 Turkish Morphology

In linguistics, the term *morphology* refers to the study of the internal structure of words. Each word is assumed to consist of one or more *morphemes*, which can be defined as the smallest linguistic unit having a particular meaning or grammatical function. One can come across morphologically simplex words, i.e. *roots*, as well as morphologically complex ones, such as compounds or affixed forms.

(1) Batı-lı-laş-tır-ıl-ama-yan-lar-dan-mış-ız
west-With-Make-Caus-Pass-Neg.Abil-Nom-Pl-Abl-Evid-A3Pl
'It appears that we are among the ones that cannot be westernized.'

[1] https://github.com/olcaytaner/TurkishMorphologicalAnalysis

Proceedings of Recent Advances in Natural Language Processing, pages 1364–1372,
Varna, Bulgaria, Sep 2–4 2019.

The morphemes that constitute a word combine in a (more or less) strict order. Most morphologically complex words are in the "ROOT-SUFFIX1-SUFFIX2-. . ." structure. Affixes have two types: (i) derivational affixes, which change the meaning and sometimes also the grammatical category of the base they are attached to, and (ii) inflectional affixes serving particular grammatical functions. In general, derivational suffixes precede inflectional ones. The order of derivational suffixes is reflected on the meaning of the derived form. For instance, consider the combination of the noun göz 'eye' with two derivational suffixes -lIK and -CI: Even though the same three morphemes are used, the meaning of a word like göz-cü-lük 'scouting' is clearly different from that of göz-lük-çü 'optician'.

Owing to its morphological properties, in Turkish, the problem of parsing and disambiguation constitutes a major challenge, as not only content words in their base forms but also functional morphemes may be the source of ambiguity. While certain affixes have clear functions/meanings, there exist others, for which the meaning can only be determined within a context. In fact, only a few derivational suffixes (which are productively used for word-formation) are semantically transparent and some derived forms are no longer considered to be compositional in that their meaning cannot be predicted from the morphemes they contain.

2.1 Allomorphy

Linguists speak of underlying representations (UR) of morphemes, which is the representation of a form in the mental lexicon, and surface representations of morphs, which concerns their exact pronunciation. While some morphemes may be realized in a single way, others have several allomorphs, i.e. phonetic variants. There are, in general, two major kinds of allomorphy with respect to their source: (i) phonologically conditioned, and (ii) morphologically or lexically conditioned.

2.1.1 Phonologically-Conditioned Allomorphy

Turkish has a rich inventory of phonological rules affecting the pronunciation of morphemes and dictating what a well-formed word in Turkish may look (or rather sound) like. There are restrictions concerning the distribution of vowel segments (V), consonants segments (Cs) or their co-occurrence. For instance, the nature of consonant clusters is highly restricted. Due to the rules dictating the so-called Vowel-Alternation, an epenthetic high vowel is inserted to break up a disallowed sequence of consonants. Morphology must respect the rules of phonology, and therefore, for instance, an impermissible cluster cannot be formed through morphological operations – an epenthetic vowel comes to rescue as in (Table 1(b)), as opposed to (Table 1(a)).

a. kedi 'cat' + 1st person Poss = kedim 'my cat'
b. ev 'house' + 1st person Poss = evim 'my house'

Table 1: Example cases showing how morphology must respect the rules of phonology.

UR	Bare-form	Accu.	Plural
a. akl 'mind'	a.kıl	ak.l-ı	a.kıl-lar
b. sırr 'secret'	sır	sır.r-ı	sır-lar

Table 2: Example words whose underlying representation ends in an impermissible cluster or a geminate.

Bare-form	Accu.	Dative	Plural
a. kas 'muscle'	kas-ı	kas-a	kas-lar
b. kasa 'safe'	kasa-yı	kasa-ya	kasa-lar

Table 3: Example Turkish words which do not contain two successive vowels.

There are some words in Turkish whose underlying representation ends in an impermissible cluster or a geminate. In the former case, vowel epenthesis takes place (Table 2(a)) whereas words of the latter kind undergo degemination (Table 2(b)), unless they are followed by a vowel which results in the resyllabification of the second consonant in the cluster ('.' indicates syllable boundary in the examples below).

There are also restrictions on neighboring vowels. Typically, Turkish words do not contain two successive vowels, except for some loanwords. Therefore, if a vowel-initial suffix is attached to a vowel-final word, a consonant (typically 'y') emerges to avoid a VV sequence, as demonstrated in (Table 3(b)).

A further phonological operation in Turkish is Vowel Harmony, which requires any vowel to agree in backness and any high vowel to agree in rounding with the preceding vowel. Respecting the rules dictating harmony, suffixes in Turkish have various allomorphs. Turkish vowel harmony

Bare-form	Accusative	Plural
a. at ‘horse’	at-ı	at-lar
b. et ‘meat’	et-i	et-ler
c. saat ‘hour, clock’	saat-i	saat-ler

Table 4: Example Turkish words which obey/does not obey vowel harmony while taking suffixes.

Bare-form	Past
kal ‘stay’	kal-dı ‘She/he/it stayed’

Table 5: Example Turkish cases with consonant harmony.

UR	Bare-form	Accu.	Plural
a. kitab ‘book’	kitap	kitab-ı	kitap-lar
b. top ‘ball’	top	top-u	top-lar

Table 6: Example Turkish words with word-final devoicing.

operates regularly on suffixes with a very few exceptions consisting of non-alternating suffixes, the most productive of which being the progressive suffix –(I)yor (e.g. when attached to git ‘to go’, we get gidiyor ‘s/he goes’, instead of *gidiyör or *gidiyir) and some words which take a front-vowel suffix even though their last vowel is back (Table 4)).

Other than Vowel Harmony, there is also Consonant Harmony in Turkish according to which oral stops and affricates agree in voicing with the preceding segment. Hence, a suffix, such as the past tense morpheme –DI, has up to 8 allomorphs when both consonant and vowel harmonies are applicable (Table 5)).

Another process relating to the voicing properties of consonants is called ‘word-final devoicing’, according to which word-final obstruents must be voiceless. Words, or rather morphemes, that are affected by this process have a voiced obstruent in the final position in their UR (as in Table 6(a)), which gets devoiced unless a vowel-initial suffix follows.

A further alternation targets the word- or rather morpheme-final ‘k’s. Dubbed ‘k-alternation’ in the literature, this process results in the replacement of morpheme-final ‘k’s with ‘ğ’ (which phonologically means a lengthening of the vowel preceding it) when they are followed by a vowel-initial suffix. K-alternation may affect roots (as in köpek ‘dog’ + ACC = köpeği instead of *köpeki) as well as suffixes (as in göz ‘eye’ + lIK + ACC forming gözlüğü instead of *gözlükü).

a. dur ‘stop, stand’ → dur-ur	bil ‘know’ → bil-ir
b. kur ‘set up’ → kur-ar	sil ‘wipe’ → sil-er

Table 7: Examples of unpredictable allomorphy.

a. it ‘push’ → it-tir	bit ‘end’ → bit-ir
b. bak ‘look’ → bak-tır	ak ‘flow, leak’ → ak-ıt

Table 8: Examples of unpredictable allomorphy for the causative suffix.

The list of phonological processes presented in this section covers some of the most frequently occurring alternations, yet it is not exhaustive. For one, there are also processes that do not have any reflection on orthography, such as alternations on vowel length. There are several others which have a narrower distribution, such as vowel reduction occurring in verbs ending in a vowel.

2.1.2 Morphologically or Lexically Conditioned Allomorphy

The crucial difference between phonologically conditioned allomorphy from other types of allomorphy is that in the former case, the alternations are predictable and apply regularly, while in the latter case the phonological features, by themselves, are not sufficient to define the environment in which the change takes place. Among the suffixes having several allomorphs, which cannot be accounted for by phonological premises only, is the aorist. While some of its forms are phonologically conditioned and thus predictable, others (Table 7(a) vs. Table 7(b)) are unpredictable from the phonological shape of the base they are attached to. A similar allomorphy is found for the causative suffix, as demonstrated in Table 8.

2.2 Inflectional Categories

Inflectional markers encode grammatical information and are category-selective. For instance, (i) Number (Singular vs. Plural); (ii) Case (nominative, accusative, dative, locative, ablative, genitive, comitative); and (iii) Possessive are encoded by inflectional suffixes (or the lack of them) on nominal stems.

Inflectional suffixes attached to verbal stems encode (i) Tense/Aspect/Modality (such as Past, Future, Aorist, Progressive, Evidential, Optative, Conditional, Ability/Possibility, Obligative etc.); (ii) Agreement (person & number); (iii) Voice (Passive, causative, reflexive, reciprocal); and (iv) Polarity (Affirmative vs. Negative).

Due to spatial restrictions, only a brief overview of morphological processes in Turkish is presented in this paper. For further information on the linguistic structure of Turkish in general, and on Turkish morphology in particular, the reader is referred to (Lewis, 1967), (Kornfilt, 1997), (Underhill, 1976), (Göksel and Kerslake, 2005), and (Erguvanlı, 2015).

3 Related Work

3.1 Available Resources

There exist several morphological analyzer resources in the Turkish NLP literature. In this section, we aim at analyzing currently available resources in terms of their usage, technology, structure, availability, and extendibility. Table 9 shows a comparison table of such resources along with our utility. Publicly unavailable resources such as widely known KIMMO-based (Karttunen et al., 1983) analyzer (Oflazer, 1994) were decidedly left out of scope of this study.

Sak et al. (2008) released the Finite-State Morphological Parser (SakMP) which uses AT&T FSM parser (Mohri, 1997). Although it is not publicly available, authors provide compiled Linux library (*.so file) upon requests. As it does not depend on any external components on runtime, researchers can call the services through the command line or Python scripting without making any installations on Linux systems. Since it is delivered as a single compiled file, its lexicon, suffixation rules and the transducer is not extendible for researchers.

TRMorph (Çöltekin, 2010) is another morphological analyzer implementation which is built upon an existing FST engine. Its latest version[2] (2.0 pre-release) uses Foma FST compiler (Hulden, 2009) which is basically a C compiler converting regular expressions to finite automata and transducers. TRMorph's lexicon files are in a raw text file format which makes them easily updatable for the researcher. It has a special regular expression based syntax (through *.xfst files) that enable researchers to update suffixation rules in compile time. Once the output file (*.fst) compiled for the platform, it can be queried with the help of Foma executables through the command line or Python scripts. The author has also introduced a web service integration with the WebLicht environment (Hinrichs et al., 2010) which allows serving TRMorph's functionality through the web interfaces (Çöltekin, 2015).

Similarly, ITU Turkish NLP Web Service (ITUWS) (Eryiğit, 2014) offers a publicly available[3] NLP user interface[4] and a web service for Turkish language which covers common tools such as tokenizer, morphological analyzer/disambiguator and dependency parser. ITUWS requires an access token on http requests in order authenticate researchers to web services. According to their paper, ITUWS wraps the morphological analyzer tool ITUMORPH (Şahin, 2013)(Şahin et al., 2013) which depends on another external tool HFST (Lindén et al., 2009) for its FST implementation. Since ITUWS is a web service-based resource, it is not suitable for tasks that require millions of analyses. Another downside of the web service-based model is the researcher could not modify any of its components such as lexicon, suffixes, and suffixation rules.

Lastly, Zemberek[5] is a popular open-source NLP framework which includes tools for Turkish such as morphological analyzer/disambiguator, tokenizer, and spell checker. It has been using as a spelling checking extension for LibreOffice[6] and the Turkish national Linux Distribution Pardus.[7] Although the project is still actively developed and maintained, its original paper is quite outdated (Akın and Akın, 2007). In documentation pages, authors note that the latest version of the library almost written from the scratch. While the original goal of the project was to abstract language specific components form the parser to support all Turkic languages (e.g., Turkmen, Azeri, Uzbek), its current focus seems on the Turkish language only. Unlike the generally accepted approach in the literature, Zemberek does not use a FST for morphological parsing. Whereas it allows developers to easily modify the lexicon through text files and the API, updating the suffixation and morphotactic rules require recompilation because of they are represented in the core Java code.

4 Core Components

Our morphological analyzer consists of five main components, namely, a lexicon, a finite state trans-

[2] https://github.com/coltekin/TRmorph

[3] http://tools.nlp.itu.edu.tr/

[4] http://tools.nlp.itu.edu.tr/MorphAnalyzer

[5] https://github.com/ahmetaa/zemberek-nlp

[6] https://extensions.libreoffice.org/extensions/zemberek-turkce-yazim-denetleyicisi

[7] https://www.pardus.org.tr/

Resource	Ours	SakMP	TRMorph	ITUWS	Zemberek
Paper	this paper	Sak et al. (2008)	Çöltekin (2010)	Eryiğit (2014)	Akın and Akın (2007)
Availability	open-source	by request	open-source	free, by request	open-source
Form	Java library	binary	Foma impl.	web service	Java library
Compile-time	Java runtime	AT&T FSM	Foma, C proc.	invisible	Java runtime
Runtime Dep.	Java runtime	no dep.	Foma, Unix tools	any REST	Java runtime
Operating Sys.	Win, OSX, Linux	Linux	Win, OSX, Linux	Any OS	Win, OSX, Linux
FST Impl.	custom Java	AT&T FSM	Foma	HFST	no FST
Lexicon	text file	embedded	text file	invisible	text file
Suffix Rules	xml file	embedded	regex, C, lexc	invisible	Java code

Table 9: Comparison of available morphological analyzer resources.

ducer, a rule engine for suffixation, a trie data structure, and a least recently used (LRU) cache.

4.1 Lexicon

For the purposes of the present study, we will assume all idiosyncratic information to be encoded in the lexicon. While phonologically conditioned allomorphy will be dealt with by the transducer, other types of allomorphy (including the ones discussed in Section 2.1.2), all exceptional forms to otherwise regular processes, as well as words formed through derivation (except for the few transparently compositional derivational suffixes) are considered to be included in the lexicon.

Table 10 shows 10 example words taken from our lexicon, where the lexicon is sorted alphabetically. Each line in the lexicon consists of the bare-form and a set of attributes separated by white space.

4.1.1 Bare-Forms

The bare-forms in the lexicon consists of nouns, adjectives, verbs, adverbs, shortcuts, etc. Each bare-form appears the same in the lexicon except verbs. Since the bare-forms of the verbs in Turkish do not have the infinitive affix 'mAk', our lexicon includes all verbs without the infinitive affix. For instance, in Table 10, verbs 'abanmak' and 'abartmak' appear as 'aban' and 'abart' respectively.

Since our morphological analyzer must support all types of texts, the bare-forms with diacritics are included in two forms, with and without diacritics. For example, noun 'rüzgâr' appear both as 'rüzgâr' and 'rüzgar'.

Special markers are included as bare-forms such as <doc>, <s>, etc.

Some compound words are included in their affixed form. For instance, 'acemlalesi' appears as it is, but not as 'acemlale'.

Foreign words, especially proper noun foreign words, are included, so that the system can easily recognize them as proper nouns. In Table 10, the words 'abbott', 'abbigail' are example foreign proper nouns. Including foreign proper nouns, there are 19,000 proper nouns in our lexicon.

From derivational suffixes, we only include words which has taken -lI, -sIz, -CI, -lIk, and -CIlIk derivational affixes. In Table 10, the bare-forms 'abacı', 'abdallık', 'abdestli' and 'abdestlilik', are included, since they have taken one or more derivational affixes listed above.

abacı CL_ISIM	aban CL_FIIL F5PR
abart CL_FIIL F5PR	abbott IS_OA
abbigail IS_OA	abdallık CL_ISIM IS_SD
abdestli IS_ADJ	abdestlilik CL_ISIM IS_SD
rüzgar CL_ISIM	rüzgâr CL_FIIL CL_ISIM

Table 10: 10 example words from our lexicon.

4.1.2 Attributes

Each bare-form has a set of attributes give in Table 11. For instance, in Table 10, 'abacı' is a noun, therefore, it includes CL_ISIM attribute. Similarly, 'abdestli' is an adjective, which includes IS_ADJ attribute. If the bare-form has homonyms with different part of speech tags, all corresponding attributes are included.

4.2 Finite State Transducer

Given a possible bare-form, depending on the possible part of speech(es) of that bare-form, the finite state transducer (FST) starts with one or more initial state. FST is responsible from state transitions, where at each transition FST (i) changes from one state to another state, (ii) appends a suffix to the current surface form to generate a new surface form, (iii) produces an output, which is the current morphological analysis of the current surface form. After a set of transitions, the current surface form will be equal to the word, for which morphological analysis sought, and if also the current state is a final state, FST will output current morphological analysis as a possible mor-

Name	Purpose
CL_ISIM, CL_FIIL, ...	Part of speech tag(s)
IS_OA	Proper noun
IS_DUP	Part of a duplicate form
IS_KIS	Abbreviation, which does not obey vowel harmony while taking suffixes.
IS_UU, IS_UUU	Does not obey vowel harmony while taking suffixes.
IS_BILEŞ	A portmanteau word in affixed form, such as 'adamotu'
IS_B_SI	A portmanteau word ending with 'sı', such as 'acemlalesi'
IS_CA	Already in a plural form, therefore can not take plural suffixes such as 'ler' or 'lar'.
IS_ST	The second consonant undergoes a resyllabification.
IS_UD, IS_UDD, F_UD	Includes vowel epenthesis.
IS_KG	Ends with a 'k', and when it is followed by a vowel-initial suffix, the final 'k' is replaced with a 'g'.
IS_SD, IS_SDD, F_SD	Final consonant gets devoiced during vowel-initial suffixation.
F_GUD, F_GUDO	The verb bare-form includes vowel reduction.
F1P1, F1P1-NOREF, ...	A verb, and depending on this attribute, the verb can (or can not) take causative suffix, factitive suffix, passive suffix etc.

Table 11: Attributes of the bare-forms

phological analysis. Depending on the number of initial states, the number of possible paths FST has sought, FST can output one or more possible morphological analyses.

In our morphological analyzer, FST is encoded in an xml file. Table 12 shows three example states from our Turkish FST xml file. <state> tag shows the properties of a state including; **name** of the state, if the state is a **start** state, if the state is a **final** state, the **pos** (part of speech) of the state if the state is a start state.

<to> tag shows the properties of state transitions from the current state including; **name** of the next state, **output** of the transducer while doing this transition, if the transition changes the part of the speech, **pos** of the produced surface form.

<with> tag has a text showing the suffix to append to the current surface form in this current state. **null** transitions are shown with '0' suffix. Similar to the <to> tag, <with> tag may have **output** of the transducer while doing this transition; if the transition changes the part of the speech, **pos** of the produced surface form.

In Table 12, first state is the *ConjunctionRoot* state, which shows the initial state for conjunctions. Since in Turkish conjunctions can not take suffixes, it is also a final state with no additional transitions. Second state is the *VerbalRoot(F4PW)* state, which shows the initial state for a specific

```
<state name="ConjunctionRoot" start="yes"
final ="yes" pos="CONJ">
</state>
<state name="VerbalRoot(F4PW)" start="yes"
final ="no" pos="VERB">
    <to name="Reciprocal" output="RECIP">
        <with>Hs</with>
    </to>
    <to name="PassiveHn">
        <with>0</with>
    </to>
</state>
<state name="NominalRootPlural" start="yes"
final ="no" pos="NOUN">
    <to name="Possessive">
        <with output="A3PL+PNON">0</with>
        <with output="A3PL+P1SG">Hm</with>
        <with output="A3PL+P2SG">Hn</with>
        <with output="A3PL+P1PL">HmHz</with>
        <with output="A3PL+P2PL">HnHz</with>
    </to>
</state>
```

Table 12: Example states from Turkish FST.

class of verbs. These verbs can take a reciprocal suffix 'Hs', a causative suffix 't', a passive suffix 'n', and null passive suffix. Third state is the *NominalRootPlural* state, which shows the initial state for root nouns already in plural form. Since they are already in plural form, the morphological outputs always start with 'A3PL' and depending on the possesive suffix, the person of the possessive is determined.

4.3 Morphotactic Rule Engine

Given the FST and a possible transition, the rule engine's job is to apply morphotactical rules to append the suffix (in the transition) to the current surface form to produce the suffixed surface form.

When the suffixes are appended to the bare-form, for the categorical exceptional cases, rule engine uses the attributes of the bare-forms given in Section 4.1.2. So, for example, if the bare-form is 'saat', and since it has an attribute IS_UU, when the accusative suffix is appended to that word, we get the surface form 'saati' (Table 4c)).

The most important function of morphotactic rule engine is to solve allomorphic cases, that is phonetic realization of metamorphemes such as 'Hs'. There are a total of four allomorphs defined in our FST. The allomorphs, the metamorphemes which use those allomorphs, and their possible phonetic realizations are given in Table 13).

There are also well known exceptions in the application of morphotactical rules, for example, when the pronouns 'bu', 'şu', 'o' get the suf-

A.morph	M.morpheme	Real.
D	DA, DAn, DH, DHk	d, t
A	Ar, CA, cAsHnA, DA	a, e
C	CA, CH, CHk	c, ç
H	cAsHnA, CH, CHk, DH	ı, i, u, ü

Table 13: Allomorphs defined in our FST and some of the metamorphemes and their realizations.

fix 'ylA', the surface form is not 'buyla', 'şuyla', 'oyla'; but 'bununla', 'şununla', 'onunla'. Another example is, when the pronoun 'ben' gets the suffix 'ya', the surface form is not 'bene' but 'bana'. The engine must also handle these kind of irregularities.

4.4 Trie Data Structure

One of the most important tasks of a morphological analyzer is to guess possible bare-forms given a surface form. The speed of the analyzer mainly depends on the number of initial bare-forms it starts with. When extraneous bare-forms are present, FST deals with unnecessary morphotactics and / or phonetic realizations, which decreases the speed of the analyzer significantly.

The naive approach of taking the k ($1 \leq k \leq$ length of the surface form) leftmost characters as possible bare-forms does not work well, since there are many irregularities. For instance, if the word 'ahenk' takes an accusative suffix, 'k' is replaced with 'g', resulting in the word 'ahengi'. In this case, we can not get the bare-form 'ahenk' by taking any leftmost characters of the word 'ahengi'.

To overcome these irregularities and also to accelerate the search for the bare-forms, we use a trie data structure in our morphological analyzer, and store all words in our lexicon in that data structure. For the regular words, we only store that word in our trie, whereas for irregular words we store both the original form and some prefix of that word. Let s^k represent the leftmost k characters of a string s, $s[m]$ represent the m'th character of a string s, and l represent the length of string s. The irregular cases occur, when the bare-form has one of the following attributes.

In these cases, we insert token t into the trie. Figure 1 shows the above cases on a trie data structure. After inserting all of the lexicon into the trie, we are ready for searching the candidate bare-forms of a given surface form. We just traverse the trie and select matched words in the trie as candidate bare-forms.

Attribute	t	Example
IS_BILEŞ	s^{l-1}	ademot
IS_B_SI	s^{l-2}	acemlale
IS_UD, IS_UDD, F_UD	$s^{l-2} + s[l] + s[l-1]$	aklı
IS_KG	s^{l-1} + 'g'	aheng
IS_SD, IS_SDD, F_SD	s^{l-1} + ('b' | 'c' | 'd' | 'ğ')	açlığ
F_GUD, F_GUDO	s^{l-1}	açıkl

Table 14: Tokens to be inserted for the selected attributes.

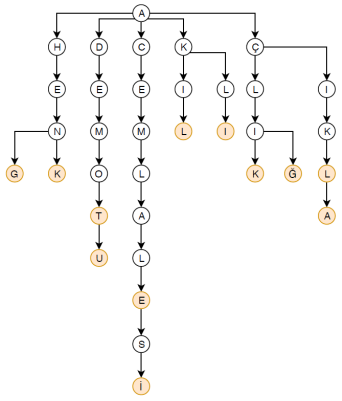

Figure 1: An example trie in our analyzer.

4.5 LRU Cache

The speed of a morphological analyzer is usually calculated on large corpora. These corpora contains millions (sometimes billions) of surface forms and as expected include many surface forms repeatedly. Since the morphological analyses of a surface form does not depend on the neighboring words, one can safely assume that, once we have extracted the morphological analyses of a word, we do not need to reextract those analyses. We can just look up the analyses of that surface form from a cache.

The idea of caching items for fast retrieval goes back nearly to the beginning of the computer science. We also use that idea and use a LRU cache for storing morphological analyses of surface forms. Before analyzing a surface form, we first look up to the cache, and if there is an hit, we just take the analyses from the cache. If there is a miss, we analyze the surface form and put the morphological analyses of that surface form in the LRU cache. As can be expected, the speed of the caching mechanism surely depends on the size of the cache. In our experiments, our cache contains up to 100K (sometimes 1M) surface forms.

5 Evaluation

5.1 Functional Evaluation

The main motivation behind our functional experiments is to test our morphological analyzer's parsing capability against known suffixation cases based on our own lexicon entries and their attributes described in the section 4.1.2. To achieve this, first, we synthetically generated test cases for 14 attributes by applying suffixes to bare-forms using our FST as a word generator. Then reversely, we tried to analyze the generated words using the same engine. After fixing some incorrectly generated cases manually, we ended up with 28,900 generated test cases in total. Lastly, we ran generated test cases on all available morphological analyzers. Table 15 shows the *passed case counts* of 14 group of test cases for each morphological analyzer. Our analyzer successfully parsed generated tests cases with 99.36% accuracy.

Test oracles for test cases slightly differ based on their group of attributes. For instance, IS_OA group test cases check whether the parser successfully yields any analysis which contains the given proper noun. This is the most simple group of test cases where its success depends heavily on the existence of such proper nouns in lexicons. Another example from group IS_UU has the test oracle "$saat \mid saatler$" meaning that at least one analysis (e.g., "saat+NOUN+A3PL+PNON+NOM") of the second string 'saatler' should contain the first string 'saat' as a bare-form. All analyzers have marked as 'passed' in this specific test case. One last special test oracle example is the IS_BI_SI attribute. This time the test oracle marks the case as passed even if the last two characters of the parsed bare-form does not match the given bare-form. This special oracle resolves the falsely failed cases caused by the incompatible bare-form structures of different analyzers as in the example of "$geceyarısı \mid geceyarıları$" where the parsed bare-form of the second string can be accepted for both 'geceyarı' or 'geceyarısı'. In this study, our test oracles do not check for any special tags (i.e., POS, MTAG) or suffixes while marking the test case as passed or not. We leave the test cases with more advanced oracles for future work.

5.2 Performance Evaluation

For the performance comparison, we used three corpora in different sizes: Milliyet (Hakkani-Tür et al., 2002), BounCorpus (Sak et al., 2008), and our corpus Gazete. All datasets are constructed from daily news websites. We excluded ITUWS and SakMP from the experiments. Since their usage models and platform (web service-based and Linux) is different, it would not yield comparable results with others. Table 16 shows final durations and number of analyses/parses of our performance tests. Number of returning parses vary depending on the authors' design choices for morphological analysis process. Parse counts are not related with the execution performances.

Our performance experiments show that our analyzer can analyze a big corpus with 37.2 million sentences in 24 minutes. Compared to the other analyzers, our tool's built-in cache mechanism leverages the memory efficiently which leads to the reduction of the analysis times.

Groups	Cases	Ours	Zemb.	TRMo.	SakMP	ITUWS
F_GUD	1,367	1,366	1,324	337	1,261	1,333
IS_BILES	1,186	1,084	547	109	260	1,186
IS_UD	176	175	151	151	132	116
IS_ST	38	37	32	25	24	14
IS_UU	347	324	265	231	243	309
IS_KG	26	26	24	24	22	25
IS_CA	415	415	348	296	326	415
F_SD	90	90	90	62	76	82
IS_UUU	10	10	7	7	6	8
IS_BI_SI	201	199	77	3	38	200
F_UD	12	12	10	6	2	2
F_GUDO	2	2	2	2	2	2
Subtotal	3,870	3,740	2,877	1,253	2,392	3,692
	100%	**96.64%**	74.34%	32.38%	61.81%	95.4%
IS_SD	5,970	5,917	2,591	2,025	3,085	-
IS_OA	19,060	19,054	14,649	15,005	12,646	-
	100%	**99.78%**	68.88%	68.04%	62.85%	-
Overall	28,900	28,716	20,117	18,283	18,123	-
	100%	**99.36%**	69.61%	63.26%	62.71%	-

Table 15: The number of passed test cases on functional evaluations, grouped by attributes.

	Ours		TRMorph		Zemberek	
Corpus (words)	Dur.	Par.	Dur.	Par.	Dur.	Par.
Milliyet (810K)	22 sec	1.2M	39 sec	-	17 sec	1.6M
Gazete (19M)	219 sec	37M	950 sec	-	330 sec	47M
BounC. (433M)	24 min	839M	161 min	-	72 min	1.1B

Table 16: Durations (Dur.) and parse counts (Par.)

6 Conclusion

In this paper, we have presented common challenges of morphological analysis task for Turkish caused by the rich morphology of the language. Then we extensively explained the internal structure of our open-source morphological analyzer toolkit which is designed to deal with such challenges. Finally, we analyzed the currently available morphological analyzer resources in the Turkish literature and reported the functional and performance-wise comparisons we have made.

References

Ahmet Afsin Akın and Mehmet Dündar Akın. 2007. Zemberek, an open source nlp framework for Turkic languages. *Structure* 10:1–5.

Çagrı Çöltekin. 2010. A freely available morphological analyzer for Turkish. In *LREC*. volume 2, pages 19–28.

Çagrı Çöltekin. 2015. Turkish nlp web services in the weblicht environment. In *Proceedings of the CLARIN Annual Conference*.

Muhammet Şahin, Umut Sulubacak, and Gülsen Eryigit. 2013. Redefinition of Turkish morphology using flag diacritics. In *Proceedings of The Tenth Symposium on Natural Language Processing (SNLP-2013), Phuket, Thailand, October*.

E.T. Erguvanlı. 2015. *The Phonology and Morphology of Turkish*. Boğaziçi University Press, İstanbul, Turkey.

Gülşen Eryiğit. 2014. ITU Turkish nlp web service. In *Proceedings of the Demonstrations at the 14th Conference of the European Chapter of the Association for Computational Linguistics*. pages 1–4.

A. Göksel and C. Kerslake. 2005. *Turkish: A Comprehensive Grammar*. Routledge, New York, USA.

Dilek Z Hakkani-Tür, Kemal Oflazer, and Gökhan Tür. 2002. Statistical morphological disambiguation for agglutinative languages. *Computers and the Humanities* 36(4):381–410.

Marie Hinrichs, Thomas Zastrow, and Erhard W Hinrichs. 2010. Weblicht: Web-based lrt services in a distributed escience infrastructure. In *LREC*.

Mans Hulden. 2009. Foma: a finite-state compiler and library. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, pages 29–32.

Lauri Karttunen et al. 1983. Kimmo: a general morphological processor. In *Texas Linguistic Forum*. volume 22, pages 163–186.

J. Kornfilt. 1997. *Turkish*. Routledge, London, UK.

G. Lewis. 1967. *Turkish Grammar*. Clarendon, Oxford, UK.

Krister Lindén, Miikka Silfverberg, and Tommi Pirinen. 2009. Hfst tools for morphology–an efficient open-source package for construction of morphological analyzers. In *International Workshop on Systems and Frameworks for Computational Morphology*. Springer, pages 28–47.

Mehryar Mohri. 1997. Finite-state transducers in language and speech processing. *Computational linguistics* 23(2):269–311.

Kemal Oflazer. 1994. Two-level description of Turkish morphology. *Literary and linguistic computing* 9(2):137–148.

Muhammet Şahin. 2013. *Itumorph: Türkçe İçin Daha Geniş Kapsamlı Ve Başarılı Bir Biçimbilimsel Çözümleyici*. Master's thesis, Fen Bilimleri Enstitüsü.

Hasim Sak. 2011. *Integrating morphology into automatic speech recognition: morpholexical and discriminative language models for Turkish*. Ph.D. thesis, PhD thesis, Boğaziçi University, Istanbul.

Haşim Sak, Tunga Güngör, and Murat Saraçlar. 2008. Turkish language resources: Morphological parser, morphological disambiguator and web corpus. In *International Conference on Natural Language Processing*. Springer, pages 417–427.

R. Underhill. 1976. *Turkish Grammar*. Cambridge University Press, Oxford, UK.