Natural Language Processing - Week03

Assist. Prof. Dr. Tuğba YILDIZ

İSTANBUL BİLGİ UNIVERSITY
Department of Computer Engineering

March 1, 2019

Morphological Parsing

- parsing means taking an input and producing some sort of linguistic structure for it.
- with morphological parsing, a word breaks down into component morphemes and building a structured representation
- Morphological Parsing applies to many affixes other than plurals (-ing, -ed, etc.)
- Surface or input form : going
- Parsed: VERB-go + GERUND-ing

Morphological Parsing

- Surface segmentation: sequence of substrings whose concatenation is the entire word
  - achievability : achiev + abil + ity
- Canonical segmentation: sequence of standardized segments
  - achievability : achieve + able + ity

Tokenization

- **A token** : is the technical name for a sequence of characters such as cars, his, :), etc.
- **Tokenization**: to break up the string into words and punctuation
- **A tokenizer** : divides a string into substrings by splitting on the specified string .

Tokenization

- NLTK!

Stemming vs Lemmatization

- **Stemming** is the process of reducing inflection in words to their root forms such as mapping a group of words to the same stem even if the stem itself is not a valid word in the Language.
- **Lemmatization**, unlike Stemming, reduces the inflected words properly ensuring that the root word belongs to the language. In Lemmatization root word is called Lemma. A lemma (is the canonical form, dictionary form, or citation form of a set of words.

Stemming vs Lemmatization

- Stemming: stripping off word endings (rule-based)
  - foxes → fox
  - going → go
  - amused → amus
- Lemmatization: mapping the word to its lemma (lexicon-based)
  - sang, sung → sing
  - going, went, goes → go

Stemming vs Lemmatization

- NLTK!

Motivation for Morphological Parsing

- Morphological Parsing is important throughout speech and language processing
  - POS tagging (Noun, verb, adj, ...)
  - Spell check
  - Information retrieval
    - Normalize verb tenses, plurals, grammar cases
    - the boy's car are different colors $\rightarrow$ the boy car be differ color
  - Machine Translation
    - Translation based on the stem

Finite-State Morphological Parsing

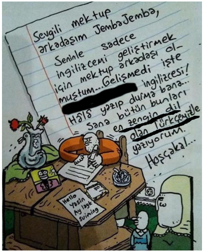
- Our goal will be to take input forms like those in first column, produce output forms in second column.
- stem and morphological features
- +N : noun
- +Sg: singular
- +pl : plural

English	
Input	Morphologically Parsed Output
cats	cat +N +PL
cat	cat +N +SG
cities	city +N +Pl
geese	goose +N +Pl
goose	goose +N +Sg
goose	goose +V
gooses	goose +V +1P +Sg
merging	merge +V +PresPart
caught	catch +V +PastPart
caught	catch +V +Past

Fig.10 Output of a morphological parse for some English words

Morphological Parsing

- Relatively simple for English. But for some languages such as Turkish, it is more difficult.



What are morphologically rich languages? - ResearchGate
https://www.researchgate.net/.../What_are_morphologically... ▾ Bu sayfanın çevirisini yap
15 May 2012 - You should definitely consider **Turkish** language (and of course other Turkic ... **Morphologically rich** languages are those which have more ...

[PDF] MORPHOLOGY-BASED AND SUB-WORD LANGUAGE ... - CmpE
https://www.cmpe.boun.edu.tr/.../Morphology-based%20an... ▾ Bu sayfanın çevirisini yap
yazan: H Sak - Alıntılanma sayısı: 21 - İlgili makaleler
ducer of the morphological parser with a language model over lex- ... **Morphologically rich** languages such as **Turkish**, Finnish, and Ara- bic present some ...

[PDF] Resources for Turkish morphological processing - CmpE
https://www.cmpe.boun.edu.tr/.../Resources%20for%20Tur... ▾ Bu sayfanın çevirisini yap
yazan: H Sak - Alıntılanma sayısı: 23 - İlgili makaleler
10 Ağu 2010 - Keywords **Turkish** language resources Á Morphological parser Á ... In **morphologically rich** languages, grammatical features and functions,.

[PS] Modeling Morphologically Rich Languages Using Split Words and ...
home.ku.edu.tr/~ebicici/publications/2009/MMRLACL.ps ▾
yazan: D Yuret - Alıntılanma sayısı: 14 - İlgili makaleler
Modeling **Morphologically Rich** Languages Using Split Words and. Unstructured Dependencies. Deniz Yuret. Koç University. 34450 Sariyer, Istanbul, **Turkey**.

[PDF] MORPHOLOGY-BASED LANGUAGE MODELING FOR TURKISH ...
citeseerx.ist.psu.edu/viewdoc/download?doi=10.1... ▾ Bu sayfanın çevirisini yap
yazan: H Sak - 2007 - İlgili makaleler
In **morphologically rich** languages like **Turkish**, we face some challenges in natural ... implementation of **Turkish** morphology based on two-level formalism.

Agglutination - Wikipedia
https://en.wikipedia.org/wiki/Agglutination ▾ Bu sayfanın çevirisini yap
Agglutination is a process in linguistic **morphology** derivation in which complex words are ... An example of such a language is **Turkish**, where for example, the word ..... In natural language processing, languages with **rich morphology** pose ...

Improving Named Entity Recognition for Morphologically Rich ...
ieeexplore.ieee.org/iel7/7031352/7033074/07033101.pdf ▾ Bu sayfanın çevirisini yap
yazan: H Demir - 2014 - Alıntılanma sayısı: 12 - İlgili makaleler
we believe it can easily be applied to other **morphologically rich** languages. Keywords—Named Entity Recognition, Word Embeddings, Skip- gram, **Turkish** NER ...

Turkish Discourse Bank: Porting a discourse annotation style to a ...
journals.linguisticsociety.org/elanguage/dad/.../2827.html ▾ Bu sayfanın çevirisini yap
yazan: D Zeyrek - 2013 - Alıntılanma sayısı: 7 - İlgili makaleler
It focuses on the challenges posed by annotating **Turkish**, a free word order language with **rich** inflectional and derivational **morphology**. It shows the usefulness ...

Morphological Parsing

- Morphemes: the small meaningful units that make up words
  - fox consists of a single morpheme: fox
  - cats consist of two morphemes: cat and -s
- Two broad classes of morphemes:
  - Stems: The core meaning-bearing units (main morpheme or the word)
  - Affixes: Bits and pieces that adhere to stems (additional meanings of various kinds)
    - prefixes : precede the stem
    - suffixes : follow the stem
    - circumfixes : do both
    - infixes : inserted inside the stem

Morphological Parsing

- There are many ways to combine morphemes to create words.
  - inflection : cat-cats, walk-walking
  - derivation : kill-killer, computerize-computerization
  - compounding : bathroom, sailboat
  - cliticization : I am/I'm, will-'ll

Morphological Parsing

- In order to build morphological parser, we will need at least the following
  - Lexicon
  - Morphotactics
  - Ortographic rules

Morphological Parsing

- In order to build morphological parser, we will need at least the following
  - Lexicon
    - List of all stems and affixes
  - Morphotactics
  - Ortographic rules

Morphological Parsing

- In order to build morphological parser, we will need at least the following
  - Lexicon
  - Morphotactics
    - the model of the morphemes ordering that explains which classes of morphemes can follow other classes of morphemesinside in a word.
    - Example: English plurals follows the nouns rather than preceeding
  - Ortographic rules

Morphological Parsing

- In order to build morphological parser, we will need at least the following
  - Lexicon
  - Morphotactics
    - Ortographic rules
    - these spelling rules are used to model the changes that occur in a word, usually when two morphemes combine
    - Example: city $\rightarrow$ cities not citys

Morphological Parsing

- A lexicon is a repository for words.
- The simplest possible lexicon would consist of an explicit list of every word of the language and proper names
- computational lexicons are usually structured with a list of each of the stem and affixes of language together with a representation of the morphotactics
- morphotactics tell us how they can fit together
- there are many ways to model morphotactics
- one of the most common is the finite-state automaton

Finite State Automaton

- Q= q0, q1,...,qN-1 : a finite set of N **states**
- $\sum$: a finite **input alphabet** of symbols
- q0: the **start state**
- F: the set of **final states**, F$\subseteq$Q
- δ(q,i) : the **transition function** or transition matrix between states. Given a state q$\in$Q and an input symbol i$\in \sum$, δ(q,i) returns a new state q'$\in$Q. δ is thus a relation form QX$\sum$ to Q.

Finite State Automaton

- For the sheeptalk automaton:
  - Q = {q0,q1,q2,q3,q4}
  - $\sum$ = {a,b,!}
  - F = {q4}
  - δ(q,i) is defined in state table
- Given a model m (such as a particular FSA), we can use L(m) to mean "the formal language characterized by m".
- L(m) = baa!, baaa!, baaaa!, baaaaa!, baaaaaa!, . . .

Finite State Automaton (FSA)

- Sheep language as any string from the following (infinite) set:
  - baa!
  - baaa!
  - baaaa!
  - ...
- Automaton models the regular expression /baa+!/.
- FSA recognizes a set of strings,

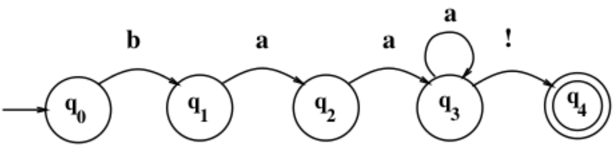


Fig.2 A finite-state automaton for talking sheep.

Finite State Automaton (FSA)

- First, a regular expression is one way of describing a finite-state automaton (FSA).
- Any regular expression can be implemented as a finite-state automaton
- Symmetrically, any finite-state automaton can be described with a regular expression.
- Second, a regular expression is one way of characterizing a particular kind of formal language called a regular language.
- Both regular expressions and finite-state automata can be used to describe regular languages
- FSA, RE and RG are all equivalent ways of describing regular languages.

Finite State Automaton (FSA)

- The automaton;
  - has five states, which are represented by nodes in the graph.
  - State 0 is the start state which we represent by the incoming arrow.
  - State 4 is the final state or accepting state, which we represent by the double circle.
  - It also has four transitions, which we represent by arcs in the graph.

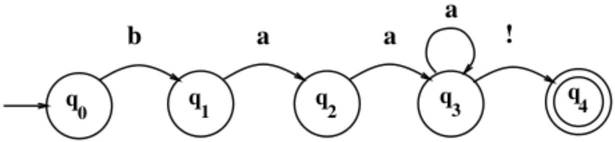


Fig.2 A finite-state automaton for talking sheep.

Finite State Automaton (FSA)

- The automaton;
  - The machine starts in the start state (q0), and iterates the following process:
    - Check the next letter of the input.
    - If it matches the symbol on an arc leaving the current state, then cross that arc, move to the next state, and also advance one symbol in the input.
    - If we are in the accepting state (q4) when we run out of input,the machine has successfully recognized an instance of sheeptalk.
    - If the machine never gets to the final state, machine rejects.

Finite State Automaton

- We can also represent an automaton with a state-transition table.
- If we're in state 0 and we see the input b we must go to state 1.
- If we're in state 0 and we see the input a or !, we fail".

	Input		
State	b	a	!
0	1	∅	∅
1	∅	2	∅
2	∅	3	∅
3	∅	3	4
4:	∅	∅	∅

Fig.3 The state-transition table for the sheeptalk example

Finite State Automaton

- Non-Deterministic FSAs:
- When we get to state 2, if we see an **a**, we do not know whether to remain in state 2 or go on to state 3.
- Another common type of non-determinism, caused by arcs that have no symbols on them (called ε-transitions)

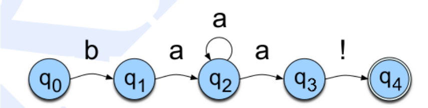


Fig.4 A non-deterministic finite-state automaton for talking sheep.

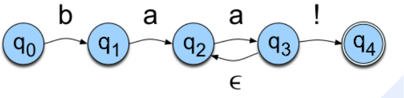


Fig.5 Another non-deterministic finite-state automaton for talking sheep.

Finite State Automaton

- Non-Deterministic FSAs:
- More than one choice at some point, we might take the wrong choice.
- solutions:
  - **Backup**: Whenever we come to a choice point, we could put a marker to mark where we were in the input and what state the automaton was in. Then if it turns out that we took the wrong choice, we could back up and try another path.
  - **Look-ahead**: We could look ahead in the input to help us decide which path to take
  - **Parallelism**: Whenever we come to a choice point, we could look at every alternative path in parallel.

Building a Finite State Lexicon

- lexicon includes regular nouns (reg-noun) that take the regular -s plural (e.g. cat, dog)

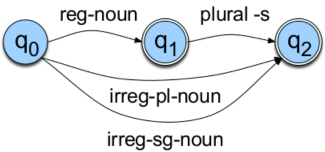


Fig.11 FSA for English nominal inflection

Building a Finite State Lexicon

- lexicon has three stem classes (reg-verb-stem, irreg-verb-stem and irreg-past-verb-form), plus four more affix classes (-ing, -ed, -ed, -s)

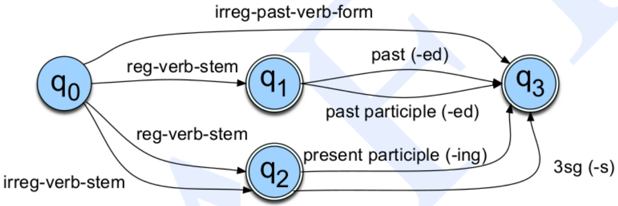


Fig.11 FSA for English verbal inflection

Building a Finite State Lexicon

- FSA will recognize the adjectives

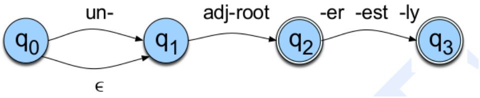
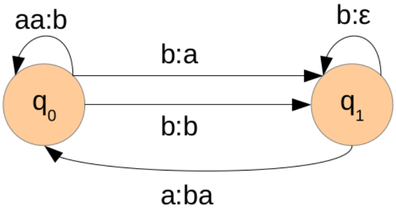


Fig.12 FSA for English adjective morphology

Finite State Transducers

- FSAs can represent the morphotactic structure of a lexicon and can be used for word recognition.
- Finite-state transducer (FST) is a type of finite automaton which maps between two sets of symbols.
- a two-tape automaton which recognizes or generates pairs of strings.
- FST is as a machine that reads one string and generates another.

Finite State Transducers



FSTs for Morphological Parsing

- In the finite-state morphology paradigm that we will use;
  - we represent a word as a correspondence between a lexical and surface levels
  - lexical level (Upper) represents a concatenation of morphemes making up a word : cat+N+PL
  - surface level (Lower) which represents the concatenation of letters which make up the actual spelling of the word : cats

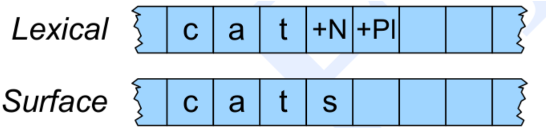


Fig.13 Surface and lexical tape

FSTs for Morphological Parsing

- goose/geese: g:g o:e o:e s:s e:e
  - Feasible pairs (e.g., o:e) vs. default pairs (g:g)

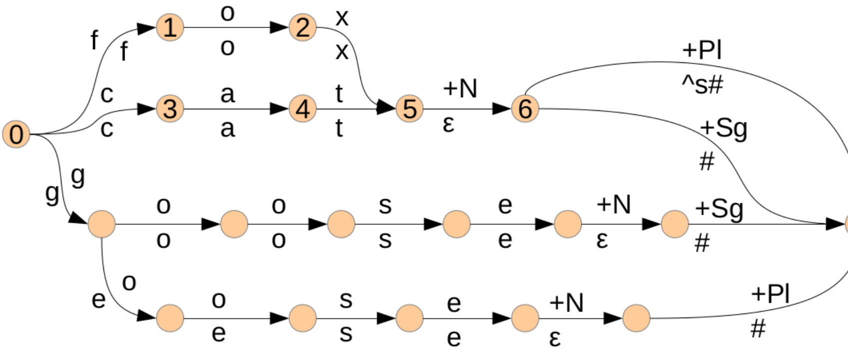


Fig.13 Surface and lexical tape

FSTs for Morpohological Parsing

- The lexical tape is composed from characters from one alphabet $\sum$
- The surface tape is composed from characters from another alphabet Δ
- two-level morphology of Koskenniemi (1983)

FSTs for Morpohological Parsing

- in two-level morphology of Koskenniemi (1983):
  - we allow each arc only to have a single symbol from each alphabet.
  - we can combine the two symbols from two different alphabet to create a new alphabet $\sum$'
  - $\sum$' is a finite alphabet of complex symbols.
  - each complex symbol is composed of an input-output pair i:o
  - one symbol i from input alphabet $\sum$
  - one symbol i from output alphabet Δ
  - $\sum' \subseteq \sum x \Delta$

FSTs for Morpohological Parsing

- $\sum = \{$ b,a,! $\}$
- $\sum' = \{$ a:a,b:b:,!:!,a:!,a:ϵ, ϵ :! $\}$
- pair symbol a : b in the transducer alphabet $\sum'$ expresses how the symbol a from one tape is mapped to the symbol b on the other tape.
- For example a: ϵ means that an a on the upper tape will correspond to nothing on the lower tape.
- output symbols include the morpheme and word boundry markers $\wedge$ and $\#$

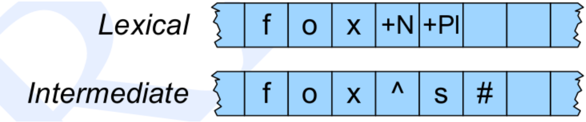


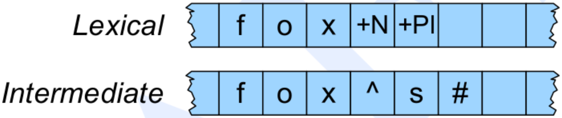
Fig.14 Lexical and intermediate tape

FSTs for Morpohological Parsing

- orthographic or spelling rules:
- incorrectly reject an input foxes and accept an input like foxs
- need to check spelling changes

Name	Description of Rule	Example
Consonant doubling	1-letter consonant doubled before *-ing*/*-ed*	beg/begging
E deletion	Silent e dropped before *-ing* and *-ed*	make/making
E insertion	e added after *-s*,*-z*,*-x*,*-ch*, *-sh* before *-s*	watch/watches
Y replacement	*-y* changes to *-ie* before *-s*, *-i* before *-ed*	try/tries
K insertion	verbs ending with *vowel* + *-c* add *-k*	panic/panicked

Fig.15 Spelling rules



FSTs for Morpohological Parsing

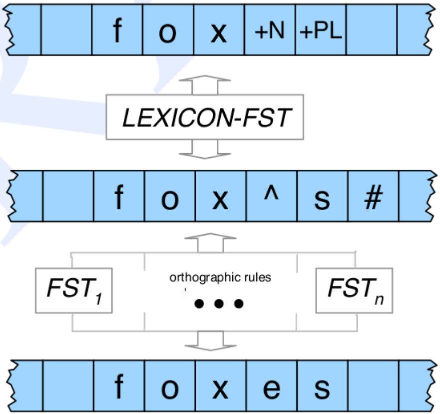


Fig.17 FST lexicons and rules

Words and Transducers

- woodchuck or woodchucks (easy!)
- fox, fish, goose, peccary!!!
- two kinds of knowledge to correctly search for singulars and plurals of these forms
  - **Orthographics rules**: English words ending in -y are pluralized by changing the -y to i and adding an -es
    - ünlü uyumu,
    - ünlü düşmesi (oğul/oğlu),
    - ünlü daralması (anla+yor/anlıyor),
    - ünsüz sertleşmesi (meslek +daş/meslektaş),
    - ünsüz yumuşaması (ağaç+a/ağaca),
    - ünsüz türemesi (his+i/hissi)
  - **Morphological rules**: fish has a null plural and plural of goose is formed by changing the vowel.

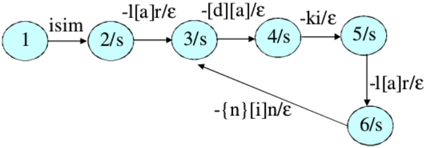
Morphological Parsing

- In order to build a morphological parser, we'll need at least the following:

1. lexicon: the list of stems and affixes, together with basic information about them (whether a stem is a Noun stem or a Verb stem, etc.).
2. morphotactics: the model of morpheme ordering that explains which classes of morphemes can follow other classes of morphemes inside a word. For example, the fact that the English plural morpheme follows the noun rather than preceding it is a morphotactic fact.
3. orthographic rules: these spelling rules are used to model the changes that occur in a word, usually when two morphemes combine (e.g., the y $-->$ ie spelling rule discussed above that changes city + -s to cities rather than citys).

Morphological Parsing

- Example in Turkish:



akıllardakilerin--> isim + l[a]r + [d][a] + ki + l[a]r + {n}[ı]n

akıl + lar + da +ki + ler + in

evlerdekilerde--> isim +l[a]r + [d][a] + ki + l[a]r + [d][a]

ev + ler + de + ki + ler + de

Fig.17 Turkish example

Morphological Parsing

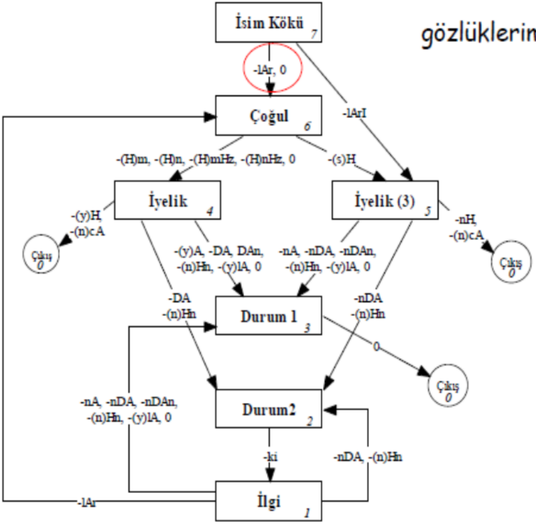


Fig.18 Turkish example

gözlüklerimden

- gözlük
- ler
- im
- den

A={a,e}
H={ı,i,u,ü}
D={d,t}

Morphological Analysis

- Word analysis is handled in four steps as;
  1. Syllabification check (e.g. YAPMAK can be typeseted YPMAK or YAPMKA)
  2. Root determination
  3. Morphological check
  4. Morphological analysis
- During these steps a dictionary of Turkish root words and set of rules for Turkish syllabus structure, morphophonemicss and morphology are used.

Morphological Analysis

- Root determination

1. the root is searched in the dictionary using maximal match algorithm
2. at first, the whole word is searched in dictionary
3. if it is found in dictionary then the word has no suffixes and its spelling is correctly
4. otherwise, remove a letter from right and serach the resulting substring
5. continue removing until finding a root
6. if no root can be found although the first letter of the word is reached (tek harf) and the word is reported as misspelled

Morphological Analysis

- Root determination

7 the maximum length substring of the word is not always its root.

8 if the root is found, the other part is checked whether they are valid for rules

- yazıldın: yazı+ldın (no suffix -ldın)

9 Root determination presents some difficulties when the root of the word is deformed.

10 For the root words which have to be deformed during certain agglutinations, a flag indicating that property is set in the dictionary.

Morphological Analysis

- Root determination
- The root of the word, ŞEHRE (to the city) must be found as ŞEHİR (city).
- In order to determine it correctly, when the substring SEHR is not found in the dictionary, considering that it may be a deformined root by vowel.
- the vowel İ is inserted between the consonants H and R, and the word ŞEHİR is searched in the dictionary.
- When it is found, the flag corresponding to vowel is checked.
- Since it is set for this word, the root of the word ŞEHRE is determied as ŞEHİR and remianing analyses are continued.
- ŞEHİR+dative case suffix

Morphological Analysis

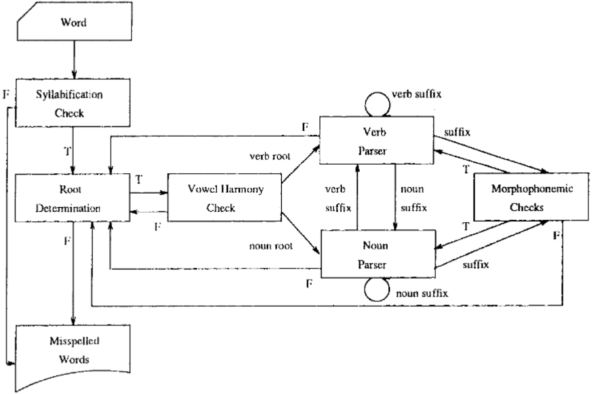
- Root determination
- b, c, d ,g/ğ $--->$ p, ç, t, k
- e.g. bacağım (bacak)
- e.g. aldığımız $--->$ aldığımı, aldığım, aldığı, aldığ, aldık, aldı, ald, alıd, alıt, alt, al

Morphological Analysis

- Morphological check:
- Turkish words obey vowel and consonant harmony rules during agglutination (ünlü uyumu vs.)
- After the root of the word is found, the rest of the word is considered as its suffixes.

Morphological Analysis

- Morphological Analysis:
- with morphological parsing, a word breaks down into component morphemes and building a structured representation



Morphological Analysis

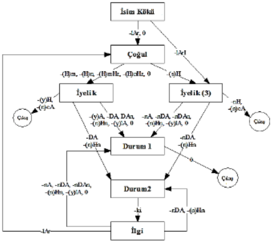


Fig.20 FST for nouns in Turkish

Morphological Analysis

Ek No	Ek	Açıklama	Örnek
1	–lAr	Çoğul	anne-ler
2	–(H)m	1. tekil kişi iyelik	anne-m
3	–(H)mHz	1. çoğul kişi iyelik	anne-miz
4	–(H)n	2. tekil kişi iyelik	anne-n
5	–(H)nHz	2. çoğul kişi iyelik	anne-niz
6	–(s)H	3. tekil kişi iyelik	anne-si
7	–lArI	1. çoğul kişi iyelik	anne-leri
8	–(y)H	-i hali	anne-yi
9	–nH	-i hali (3.t.k. iyelikten sonra)	anne-si-ni
10	–(n)Hn	tamlama	anne-nin
11	–(y)A	-e hali	anne-ye
12	–nA	-e hali (3.t.k. iyelikten sonra)	anne-si-ne
13	–DA	-de hali	anne-de
14	–nDA	-de hali (3.t.k. iyelikten sonra)	anne-si-nde
15	–DAn	-den hali	anne-den
16	–nDAn	-den hali (3.t.k. iyelikten sonra)	anne-sin-den
17	–(y)lA	birliktelik	anne-yle
18	–ki	ilgi	annem-de-ki
19	–(n)cA	görelik	annem-ce

Fig.21 Noun suffixes in Turkish

Parse

- Let's try **Sak Parser** (Haşim Sak, 2008)

References

- Speech and Language Processing (3rd ed. draft) by D. Jurafsky & J. H. Martin (web.stanford.edu)
- http://www.bmbb.info/dosya/dergi/6_3.pdf
- https://www.aclweb.org/anthology/C/C92/C92-1010.pdf
- Haşim Sak, Tunga Güngör, and Murat Saraçlar: Resources for Turkish Morphological Processing. Language Resources and Evaluation, Vol. 45, No. 2, pp. 249–261, 2011.