NATURAL LANGUAGE PROCESSING

LESSON 2: TEXT NORMALIZATION, LEMMATIZATION, PARSING

OUTLINE

- Text Normalization
  - Tokenizing wordS
  - Normalization of word formats
  - Segmenting sentences
- Lemmatization
  - Lemma and Stem
  - Lemmatization with similarity function
  - Lemmatization with FSA
- Parsing
  - Parsing tree
  - Ambiguity

WHAT IS TEXT NORMALIZATION?

- Normalization is a process that converts a list of words to a more uniform sequence. This is useful in preparing text for later processing
- Normalizing text before storing or processing it, allows for separation of concerns, since input is guaranteed to be consistent before operations are performed on it.

TEXT NORMALIZATION

Every NLP task needs to do text normalization:

1. Tokenizing words in running text
2. Normalization of word formats
3. Segmenting sentences in running text

TEXT NORMALIZATION

There are two important concepts to know:

- **Lemma**: an element of the vocabulary as listed in dictionary:

  kale

- **Token**: an instance of that type in running text.
  Can be found in any wordform in the sentences:

  kale, kaleyi, kaleden

WHAT IS TOKEN? –An English example

- ‘‘Token is an individual occurrence of a symbol or string in speech or writing’’

- Tokens of the definition: Token, is, an, individual, occurrence, of, a, symbol, or, string, in, speech, or, writing

TEXT NORMALIZATION

Mağaradaki oyuklar sanki yontma taş devrinden kalmış gibiydi.

- Tokenizing words in running text
  - Mağaradaki, oyuklar, sanki, yontma taş devrinden, kalmış, gibiydi
- Normalizing word formats
  - Mağara, oyuk, sanki, yontma taş devri, kal, gibi,

TEXT NORMALIZATION

|*T*| = the number of tokens

V = vocabulary (set of lemmas)

|*V*| = the size of the vocabulary

$|V| > O(|T|^{1/2})$ .....(Church and Gale, 1990)

	|T|	|V|
Switchboard phone conversations	2.4 million	20 thousand
Shakespeare	884,000	31 thousand
Google N-grams	1 trillion	13 million

TOKENIZATION EXAMPLE IN UNIX - 1

(Inspired by Ken Church's UNIX for Poets.)

Given a text file, output the word tokens and their frequencies

```
tr -sc 'A-Za-z' '\n' < shakes.txt
      | sort
      | uniq -c
1945 A
  72 AARON
  19 ABBESS
   5 ABBOT
  ... ...
```

Change all non-alpha to newlines

Sort in alphabetical order

Merge and count each type

TOKENIZATION EXAMPLE IN UNIX - 2

- Justext (a kind of text process program) program is used to clean the html tags from the html pages.
- After that, punctuation signs are cleaned in Linux:

```
sed -i -e 's/'"'"'//g' -e 's/’//g' -e 's/:/./g' -e 's/“//g' -e 's/”//g' -e 's/"//g'
-e 's/*//g' -e 's/-//g' -e 's/[(][^)]*[)]//g' -e 's/\.\.\./\./g' -e 's/‘//g'  -e
's/‘‘//g' -e 's/…//g' -e 's/´//g'  -e 's/¦//g'  -e 's/¨//g' -e 's/,//g'
Complete_Wget.txt
```

- The clean sentences are tokenized in a Java program to obtain tokens (words)

TOKENIZATION: LANGUAGE ISSUES

Öğrenciler'in defteri → Öğrenci Öğrenciler Öğrenciler'in?

niçin → ne için? niçin?

Ural-Altay dil grubu → Ural-Altay? Ural Altay ?

Birinci Dünya Savaşı → one token, two, three?

LEMMATIZATION

- **Lemmatization** is to reduce inflections or variant forms to base form
  - *am, is, are → be*
  - *car, cars, car's, cars' → car*

the boy's cars are different colors → the boy car be different color

- In Turkish, **Lemmatization** is a bit more difficult problem because of the agglutinative structure
- Used especially in Machine translation
  - In Turkish koşarım (I run), koşarsın (You run) have same lemma as koş (run)

MORPHOLOGY

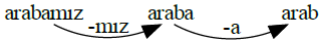
- **Morphology** is the study of the internal structure of words and forms
- **Morphemes**:
  - The small meaningful units that make up words
    - **Stems**: The core meaning-bearing units (göz)
    - **Affixes**: Bits and pieces that adhere to stems (-lük)
      - They form the word "gözlük"
      - Often with grammatical functions

STEMMING

- Reduce terms to their stems in information retrieval
- *Stemming* is crude chopping of affixes
  - language dependent
  - e.g., ***boyama, boyalı, boyacı*** all reduced to ***boya***.

Arabamız kelimesinde arab köküne kadar gidebiliyor ancak asıl gövde arabadır.

Arab kelime arab kök kadar git ancak asıl gövde arab



LEMMATIZATION OF TURKISH

Lemmatization of Turkish words can be performed by using:

- a Similarity Function
- a Finite State Automaton

LEMMATIZATION BY SIMILARITY FUNCTION

Similarity Function:

- A normalized similarity function for Turkish words which compares two words and calculates a score according to their similarity.
- Consonant voicing changes and Consonant drops are also handled in this function to increase accuracy.
- The function always produces a normalized value in the range [0 1]

LEMMATIZATION BY SIMILARITY FUNCTION

Similarity Function:

- The length of word is planned for 25 characters, and "ε" value is chosen as 1/25=0.04
- Inspired by Hamming Distance

$$S(w,u) = \frac{\sum_{i \le n}(w_i == u_i)}{n} - (\varepsilon * (m-n)) \qquad (1)$$

$$(1-(\varepsilon*9)<p<1-(\varepsilon*8)) \quad (2)$$

1	2	3	4	5	6	7	8	9	
a	l	d	ı	ğ	ı	m			n=7
a	l	d	ı	k	l	a	r	ı	m=9

- Assuming the longest affix in Turkish Language as maximum 8 characters, a reasonable threshold value is defined as: p=0.65

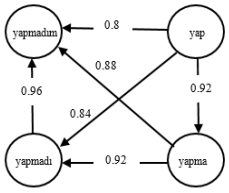
17

LEMMATIZATION BY SIMILARITY FUNCTION

Similarity Function:

- Similarity of a lemma with one of its inflected surface form can be computed by multiplying each edge of lemma and inflected word
- The similarity values these two words with their common lemma "yap" is approximately equal to the multiplied of values as shown in (3).
- The parameters (w_L, w_1, w_2, w_3) in (3) represent the words ("yap", "yapma", "yapmadı", "yapmadım")

$$S(w_L,w_2) \cong S(w_L,w_1)*S(w_1,w_2) \qquad (3)$$
$$S(w_L,w_3) \cong S(w_L,w_1) * S(w_1,w_2) * S(w_2,w_3)$$



18

LEMMATIZATION BY FSA

The FSAs used in affix validation module are:

- Noun suffixes FSA (class#1)
- Nominal verb suffixes FSA (class#2)
- Tense & Person verb suffixes FSA (class#3)
- Verb inflectional suffixes FSA (class#4)

LEMMATIZATION BY FSA

Noun Suffixes (#class 1):

Suffix No	Suffix	Description for English language	Example
1	-lAr	Plural	gemi-ler
2	-(H)m	1st single person possessive	gemi-m
3	-(H)mHz	1st plural person possessive	gemi-miz
4	-(H)n	2nd single person possessive	gemi-n
5	-(H)nHz	2nd plural person possessive	gemi-niz
6	-(s)H	3rd single person possessive	gemi-si
7	-lArI	1st plural person possessive	gemi-leri
8	-(y)H	accusative case	gemi-yi
9	-nH	accusative case after possessive suffix added to lemma ends with vowel	gemi-si-ni
10	-(n)Hn	possessive	gemi-nin
11	-(y)A	dative case	gemi-ye
12	-nA	dative case after possessive suffix added to lemma ends with vowel	gemi-si-ne
13	-DA	preposition (in)	gemi-de
14	-nDA	preposition (in) possessive suffix added to lemma ends with vowel	gemi-si-nde
15	-DAn	preposition (from)	gemi-den
16	-nDAn	preposition (from) possessive suffix added to lemma ends with vowel	gemi-sin-den
17	-(y)lA	preposition (with)	gemi-yle
18	-ki	possessive	gemim-de-ki
19	-(n)cA	equative	gemim-ce

LEMMATIZATION BY FSA

Nominal Verb Suffixes (#class 2):

Suffix No	Suffix	Description	Example
1	–(y)Hm	1st single person	çalışkan-ım
2	–sHn	2nd single person	çalışkan-sın
3	–(y)Hz	1st plural person	çalışkan-ız
4	–sHnHz	2nd plural person	çalışkan-sınız
5	–lAr	3rd plural person	çalışkan-lar
6	-m	1st single person	çalışkan-dı-m
7	-n	2nd single person	çalışkan-dı-n
8	-k	1st plural person	çalışkan-dı-k
9	–nHz	2nd plural person	çalışkan-dı-nız
10	–DHr	to be	çalışkan-dı-r
11	–cAsInA	adverbs of manner	çalışkan-mış-çasına
12	–(y)DH	verbal noun past tense	çalışkan-dı
13	–(y)sA	subjunctive mood	çalışkan-sa
14	–(y)mHş	past perfect tense	çalışkan-mış
15	–(y)ken	adverb of time	çalışkan-ken

LEMMATIZATION BY FSA

Tense Person Verb Suffixes (#class 3):

Suffix No	Suffix	Description	Example
1	–(y)Hm	1st single person	araştır-ıyor-um
2	–sHn	2nd single person	araştır-ıyor-sun
3	–(y)Hz	1st plural person	araştır-ıyor-uz
4	–sHnHz	2nd plural person	araştır-ıyor-sunuz
5	–lAr	3rd plural person	araştır-ıyor-lar
6	-mHş	past perfect tense	araştır-mış
7	–(y)AcAk	future tense	araştır-acak
8	–(H)r	simple present tense	görü-r
9	–Ar	simple present tense	koşa-r
10	–(H)yor	present tense	araştır-ıyor
11	–mAktA	present perfect continuous tense	araştır-makta
12	–mAlI	modal verb	araştır-malı
13	-m	1st single person	araştır-dı-m
14	-n	2nd single person	araştır-dı-n
15	-k	1st plural person	araştır-dı-k
16	–nHz	2nd plural person	araştır-dı-nız
17	–DH	past tense	araştır-dı
18	–sA	subjunctive mood	araştır-sa

LEMMATIZATION BY FSA

Verb Inflectional Suffixes (#class 4):

Suffix No	Suffix	Description	Example
1	-m	1st single person	koş-ma-m
2	–zsIn	2nd single person	koş-ma-zsın
3	–z	3rd single person	koş-ma-z
4	–yIz	1st plural person	koş-ma-yız
5	–zsInIz	2nd plural person	koş-ma-zsınız
6	-zlAr	3rd plural person	koş-ma-zlar
7	-mA	negative	koş-ma
8	-(y)AmA	negative	koş-ama
9	-(y)Adur	verbs of continuity	koş-adur
10	-(y)Hver	verb of quickness	koş-uver
11	-(y)Agel	verbs of continuity	koş-agel
12	-(y)Agör	verbs of continuity	oku-yagör-sün
13	-(y)Abil	verb of ability	koş-abil
14	-(y)Ayaz	verb of imminence	düş-eyaz-dı-m
15	-(y)Akal	verbs of continuity	baka-kal
16	-(y)Akoy	verbs of continuity	alı-koy-sun
17	-mAk	infinitive	koş-mak
18	-(y)HcH	task affix	koş-ucu

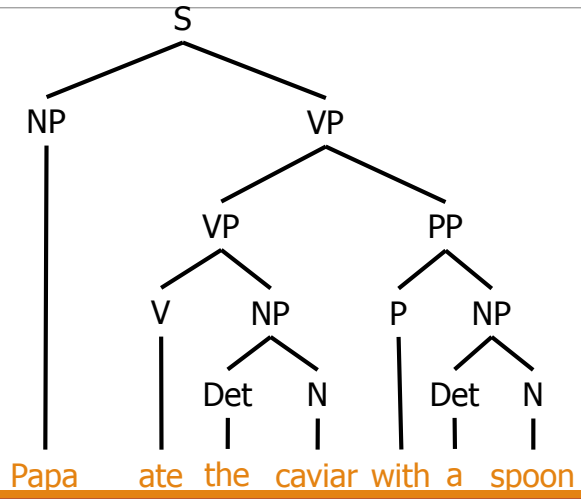
23

WHAT IS PARSING?

- Parsing is a process of analyzing a sentence by taking each word and determining its structure from its constituent parts. Parsing process makes use of two components: a parser and a grammar.
- Parser is a procedural component and is nothing but a computer program. Grammar is a declarative component.
- Both the grammar and parser depend on the grammar formalism

WHAT IS PARSING?

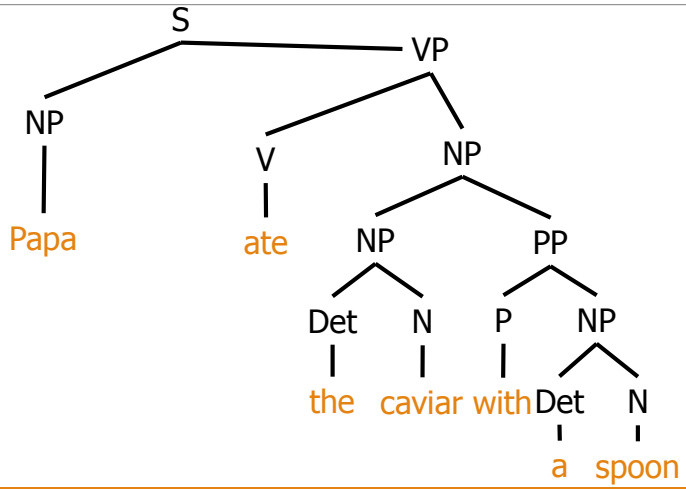
S → NP VP
NP → Det N
NP → NP PP
VP → V NP
VP → VP PP
PP → P NP



NP → Papa
N → caviar
N → spoon
V → spoon
V → ate
P → with
Det → the
Det → a

AMBIGUITY

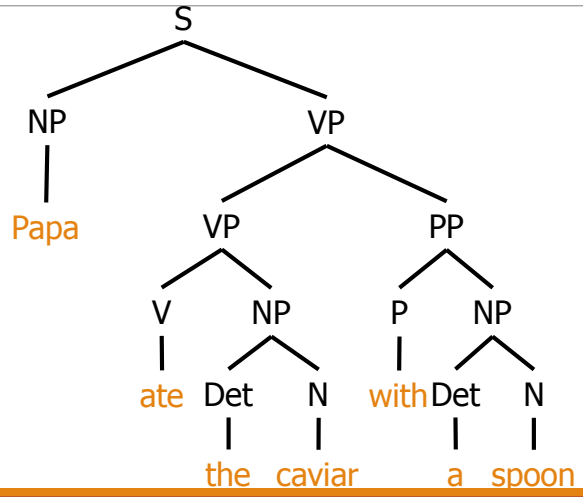
S → NP VP
NP → Det N
NP → NP PP
VP → V NP
VP → VP PP
PP → P NP



NP → Papa
N → caviar
N → spoon
V → spoon
V → ate
P → with
Det → the
Det → a

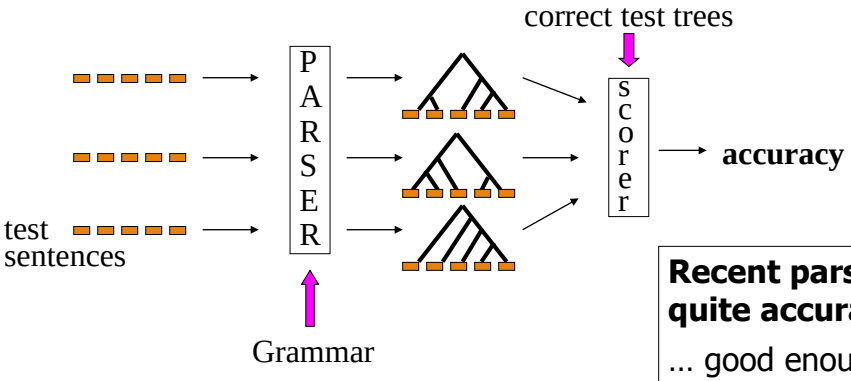
AMBIGUITY

S → NP VP
NP → Det N
NP → NP PP
VP → V NP
VP → VP PP
PP → P NP



NP → Papa
N → caviar
N → spoon
V → spoon
V → ate
P → with
Det → the
Det → a

THE PARSING PROBLEM



Recent parsers quite accurate

... good enough to help a range of NLP tasks!

APPLICATIONS OF PARSING

- Machine translation (Alshawi 1996, Wu 1997, ...)
- Indexing for information retrieval (Woods 1997)
- Information extraction (Hobbs 1996)
- Speech synthesis from parses (Prevost 1996)
- Speech recognition using parsing (Chelba et al 1998)