Parts of Speech Part 1 ICS 482 Natural Language Processing

Lecture 9: Parts of Speech
Part 1

Husni Al-Muhtaseb

NLP Credits and Acknowledgment

These slides were adapted from presentations of the Authors of the book

SPEECH and LANGUAGE PROCESSING:

An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition

and some modifications from presentations found in the WEB by several scholars including the following

NLP Credits and Acknowledgment

If your name is missing please contact me
muhtaseb
At
Kfupm.
Edu.
sa

NLP Credits and Acknowledgment

Husni Al-Muhtaseb
James Martin
Jim Martin
Dan Jurafsky
Sandiway Fong
Song young in
Paula Matuszek
Mary-Angela Papalaskari
Dick Crouch
Tracy Kin
L. Venkata Subramaniam
Martin Volk
Bruce R. Maxim
Jan Hajič
Srinath Srinivasa
Simeon Ntafos
Paolo Pirjanian
Ricardo Vilalta
Tom Lenaerts

Heshaam Feili
Björn Gambäck
Christian Korthals
Thomas G. Dietterich
Devika Subramanian
Duminda Wijesekera
Lee McCluskey
David J. Kriegman
Kathleen McKeown
Michael J. Ciaraldi
David Finkel
Min-Yen Kan
Andreas Geyer-Schulz
Franz J. Kurfess
Tim Finin
Nadjet Bouayad
Kathy McCoy

Khurshid Ahmad
Staffan Larsson
Robert Wilensky
Feiyu Xu
Jakub Piskorski
Rohini Srihari
Mark Sanderson
Andrew Elks
Marc Davis
Ray Larson
Jimmy Lin
Marti Hearst
Andrew McCallum
Nick Kushmerick
Mark Craven
Chia-Hui Chang
Diana Maynard
James Allan

Martha Palmer
julia hirschberg
Elaine Rich
Christof Monz
Bonnie J. Dorr
Nizar Habash
Massimo Poesio
David Goss-Grubbs
Thomas K Harris
John Hutchins
Alexandros Potamianos
Mike Rosner
Latifa Al-Sulaiti
Giorgio Satta
Jerry R. Hobbs
Christopher Manning
Hinrich Schütze
Alexander Gelbukh
Gina-Anne Levow
Guitao Gao
Qing Ma

Previous Lectures

- Pre-start questionnaire
- Introduction and Phases of an NLP system
- NLP Applications - Chatting with Alice
- Finite State Automata & Regular Expressions & languages
- Deterministic & Non-deterministic FSAs
- Morphology: Inflectional & Derivational
- Parsing and Finite State Transducers
- Stemming & Porter Stemmer
- 20 Minute Quiz
- Statistical NLP – Language Modeling
- N Grams
- Smoothing and NGram: Add-one & Witten-Bell

Today's Lecture

- Return Quiz1
- Witten-Bell Smoothing
- Part of Speech

Return Quiz

- Statistics and grades are available at course web site
- Sample Solution is also posted
- Check the sample solution and if you have any discrepancy write your note on the top of the quiz sheet and pass it to my office within 2 days.

Quiz1 Distribution

Distribution for Quiz1

Statistics: Quiz1

Graded out of: 28.0 | Highest grade: 23.0 | Mean grade: 14.3 | Standard deviation: 5.1

Number of records: 14 | Lowest grade: 8.0 | Median grade: 14.0

Score Range	Frequency
[0, 2.8)	
[2.8, 5.6)	
[5.6, 8.4)	3
[8.4, 11.2)	2
[11.2, 14)	
[14, 16.8)	5
[16.8, 19.6)	1
[19.6, 22.4)	2
[22.4, 25.2)	1
[25.2, 28)	
[28]	

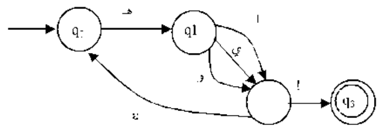
Quiz1 Sample Solution

ICS 482 Natural Language Processing (062) | Quiz #1 | Tuesday, March 11, 2007
Husni Al-Muhtaseb

Name:	ID:	Points /28

Question 1: [6 points] Draw an FSA to represent a laughing machine. The laughing machine should recognize sequences of ها, هو, and هي followed by !. It should also recognize any mix of them. Assume separate letters; i.e. هـ, ا, و , ي. The symbol "!" takes the machine to a final state.

Answer:

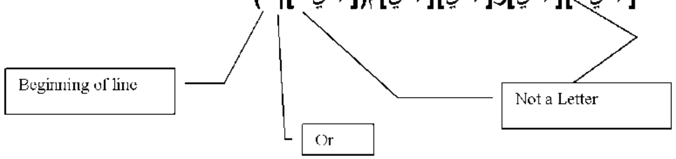


Question 2: [6 points] Write a regular expression to represent the above laughing machine.

Answer: /(ه [ا و ي])–!/

Question 3: [6 points] Write a regular expression to represent all Arabic words of the pattern مفعول. The expression should represent ***all*** strings like مكتوب, مرسوم, and so on. Avoid errors by minimizing both positive and negative errors.

(^|[^ء-ي])م[ء-ي][ء-ي]و[ء-ي][^ء-ي]

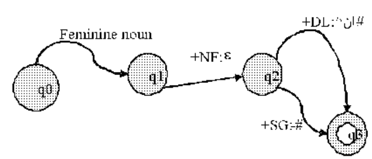


Question 4: [10 points] Study the following table for some singular and dual Arabic feminine names:

What Finite Sate Transducers do we need to accept an Arabic feminine singular name and replace it by its correspondent dual name as in the shown examples?

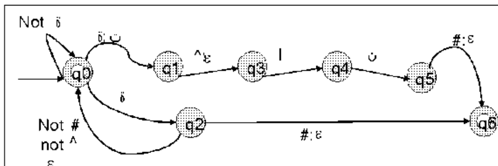
Dual	Singular
طاولتان	طاولة
شجرتان	شجرة
فاطمتان	فاطمة
ساعتان	ساعة

We might need two FSTs; one for capturing morphotactical rules and the other for capturing orthographic Rules (or spell changes). In our example, we notice that to change from singular to dual we add the two letters Alef and Ta (ان) at the end of the word. This procedure could be used in capturing morphotactical rules. To capture orthographic rules in our example we have to have an FST to change the letter Ta marbotah (ة) to open Ta (ت).



Possible FST to capture morphotactical rules (Alef and Noon (ن) attachment)

Of course, we need to replace "Feminine Noun" with every Feminine noun representation in the lexicon.



A partial possible FST to capture orthographical rules (to change Ta marbotah (ة) to open Ta (ت))

Smoothing and N-grams

- **Witten-Bell Smoothing**
  - equate zero frequency items with frequency 1 items
  - use frequency of things seen once to estimate frequency of things we haven't seen yet
  - *smaller impact than Add-One*
- **Unigram**
  - a zero frequency word (unigram) is "an event that hasn't happened yet"
  - count the number of words (T) we've observed in the corpus (Number of types)
  - p(w) = T/(Z*(N+T))
    - w is a word with zero frequency
    - Z = number of zero frequency words
    - N = size of corpus

Distributing

- The amount to be distributed is

$$\frac{T}{N+T}$$

- The number of events with count zero

$$Z$$

- So distributing evenly gets us

$$\frac{1}{Z}\frac{T}{N+T}$$

Distributing Among the Zeros

- If a bigram "w_x w_i" has a zero count

Number of bigram types starting with wx

$$P(w_i \mid w_x) = \frac{1}{Z(w_x)} \frac{T(w_x)}{N(w_x) + T(w_x)}$$

Number of bigrams starting with wx that were not seen

Actual frequency (count)of bigrams beginning with wx

Smoothing and N-grams

- **Bigram**
  - $p(w_n|w_{n-1}) = C(w_{n-1}w_n)/C(w_{n-1})$ (*original*)
  - $p(w_n|w_{n-1}) = T(w_{n-1})/(Z(w_{n-1})*(T(w_{n-1})+N))$ for zero bigrams (*after* Witten-Bell)
    - $T(w_{n-1})$ = number of bigrams beginning with w_{n-1}
    - $Z(w_{n-1})$ = number of unseen bigrams beginning with w_{n-1}
    - $Z(w_{n-1})$ = total number of possible bigrams beginning with w_{n-1} minus the ones we've seen
    - $Z(w_{n-1}) = V - T(w_{n-1})$
  - $T(w_{n-1})/ Z(w_{n-1}) * C(w_{n-1})/(C(w_{n-1})+ T(w_{n-1}))$
    - estimated zero bigram frequency
  - $p(w_n|w_{n-1}) = C(w_{n-1}w_n)/(C(w_{n-1})+T(w_{n-1}))$
    - for non-zero bigrams (*after* Witten-Bell)

Smoothing and N-grams

- **Witten-Bell Smoothing**
  - use frequency (count) of things seen once to estimate frequency (count) of things we haven't seen yet
- **Bigram**
  - $T(w_{n-1}) / Z(w_{n-1}) * C(w_{n-1}) / (C(w_{n-1}) + T(w_{n-1}))$ estimated zero bigram frequency (count)
    - $T(w_{n-1})$ = number of bigrams beginning with w_{n-1}
    - $Z(w_{n-1})$ = number of unseen bigrams beginning with w_{n-1}

	I	want	to	eat	Chinese	food	lunch
I	8	1087	0	13	0	0	0
want	3	0	786	0	6	8	6
to	3	0	10	860	3	0	12
eat	0	0	2	0	19	2	52
Chinese	2	0	0	0	0	120	1
food	19	0	17	0	0	0	0
lunch	4	0	0	0	0	1	0

Remark:
smaller changes

	I	want	to	eat	Chinese	food	lunch
I	7.785	1057.763	0.061	12.650	0.061	0.061	0.061
want	2.823	0.046	739.729	0.046	5.647	7.529	5.647
to	2.885	0.084	9.616	826.982	2.885	0.084	11.539
eat	0.073	0.073	1.766	0.073	16.782	1.766	45.928
Chinese	1.828	0.011	0.011	0.011	0.011	109.700	0.914
food	18.019	0.051	16.122	0.051	0.051	0.051	0.051
lunch	3.643	0.026	0.026	0.026	0.026	0.911	0.026

ICS 482 Natural Language Understanding

Lecture 9: Parts of Speech
Part 1

Husni Al-Muhtaseb

Parts of Speech

- Start with eight basic categories
  - Noun, verb, pronoun, preposition, adjective, adverb, article, conjunction
- These categories are based on morphological and distributional properties (not semantics)
- Some cases are easy, others are not

Parts of Speech

- Two kinds of category
  - Closed class
    - Prepositions, articles, conjunctions, pronouns
  - Open class
    - Nouns, verbs, adjectives, adverbs

Part of Speech

- Closed classes
  - Prepositions: on, under, over, near, by, at, from, to, with, etc.
  - Determiners: a, an, the, etc.
  - Pronouns: she, who, I, others, etc.
  - Conjunctions: and, but, or, as, if, when, etc.
  - Auxiliary verbs: can, may, should, are, etc.
  - Particles: up, down, on, off, in, out, at, by, etc.
- Open classes:
  - Nouns:
  - Verbs:
  - Adjectives:
  - Adverbs:

Part of Speech Tagging

- Tagging is the task of labeling (or tagging) each word in a sentence with its appropriate part of speech.
- The representative put chairs on the table.
- The[AT] representative[NN] put[VBD] chairs[NNS] on[IN] the[AT] table[NN].
- Tagging is a case of limited syntactic disambiguation. Many words have more than one syntactic category.
- Tagging has limited scope: we just fix the syntactic categories of words and do not do a complete parse.

Part of Speech Tagging

- Associate with each word a lexical tag
  - 45 classes from Penn Treebank
  - 87 classes from Brown Corpus
  - 146 classes from C7 tagset (CLAWS system)

Penn Treebank

- Large Corpora of 4.5 million words of American English
  - POS Tagged
  - Syntactic Bracketing
- : http://www.cis.upenn.edu/~treebank
  - Visit this site!

Penn Treebank

Description	Tagged for Part-of-Speech	Skeletal Parsing
	(Tokens)	(Tokens)
Dept. of Energy abstracts	231,404	231,404
Dow Jones Newswire stories	3,065,776	1,061,166
Dept. of Agriculture bulletins	78,555	78,555
Library of America texts	105,652	105,652
MUC-3 messages	111,828	111,828
IBM Manual sentences	89,121	89,121
WBUR radio transcripts	11,589	11,589
ATIS sentences	19,832	19,832
Brown Corpus, retagged	1,172,041	1,172,041
Total:	**4,885,798**	**2,881,188**

POS Tags from Penn Treebank

Tag	Description	Example	Tag	Description	Example
CC	Coordin. Conjunction	*and, but, or*	NNS	Noun, plural	*llamas*
CD	Cardinal number	*one, two, three*	NNP	Proper noun, singular	*IBM*
DT	Determiner	*a, the*	NNPS	Proper noun, plural	*Carolinas*
EX	Existential 'there'	*there*	PDT	Predeterminer	*all, both*
FW	Foreign word	*mea culpa*	POS	Possesive ending	*'s*
IN	Preposition/sub-conj	*of, in, by*	PP	Personal pronoun	*I, you, he*
JJ	Adjective	*yellow*	PP$	Possesive pronoun	*your, one's*
JJR	Adjective, comparative	*bigger*	RB	Adverb	*quickly, never*
JJS	Adjective, superlative	*wildest*	RBR	Adverb, comparative	*faster*
LS	List item marker	*1, 2, One*	RBS	Adverb, superlative	*fastest*
MD	Modal	*can, should*	RP	Particle	*up, off*
NN	Noun, singular or mass	*llama*	SYM	Symbol	+, %, &

Distribution

- Parts of speech follow the usual behavior
  - Most words have one part of speech
  - Of the rest, most have two
  - The rest
    - A small number of words have lots of parts of speech
    - Unfortunately, the words with lots of parts of speech occur with high frequency

What do POS Taggers do?

- POS Tagging
  - Looks at each word in a sentence
  - And assigns tag to each word
    - For example: *The man saw the boy.*

the-DET man-NN saw-VPAST the-DET boy-NN

Part of Speech Tagging

Some examples:

	The	students	went	to	class
	DT	NN	VB	P	NN
	Plays	well	with	others	
	VB	ADV	P	NN	
*	NN	NN	P	DT	
	Fruit	flies	like	a	banana
	NN	NN	VB	DT	NN
	NN	VB	P	DT	NN
?	NN	NN	P	DT	NN
*	NN	VB	VB	DT	NN

Sets of Parts of Speech: Tagsets

- There are various standard tagsets to choose from; some have a lot more tags than others
- The choice of tagset is based on the application
- Accurate tagging can be done with even large tagsets

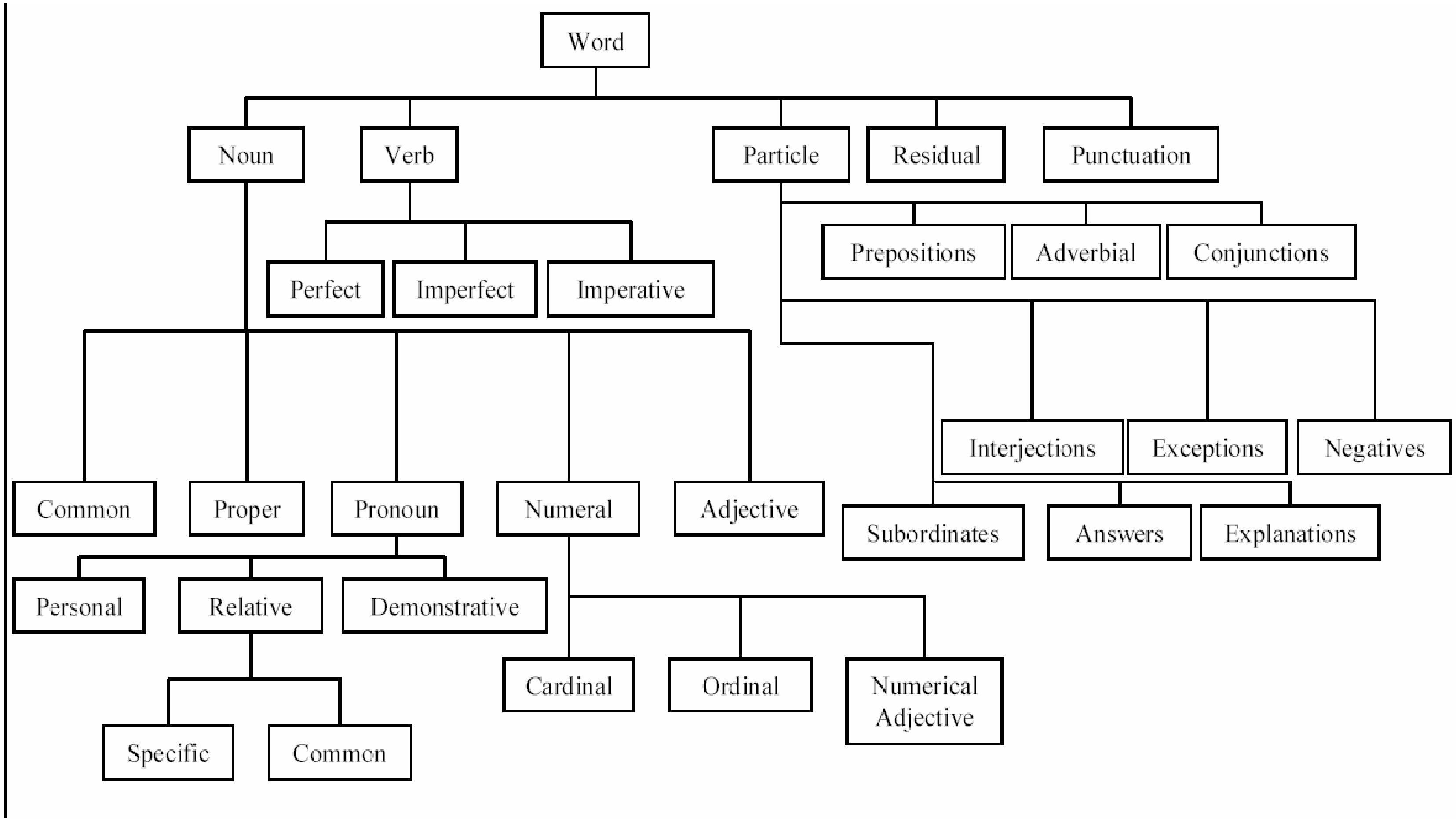
Tagging

- Part of speech tagging is the process of assigning parts of speech to each word in a sentence… Assume we have
  - A tagset
  - A dictionary that gives you the possible set of tags for each entry
  - A text to be tagged
  - A reason?

Arabic Tagging

- Shereen Khoja
  - Computing Department
  - Lancaster University
- Saleh Al-Osaimi
  - School of Computing
  - University of Leeds

Tagset Hierarchy used for Arabic



POS Tagging

- Most words are unambiguous
- Many of the most common English words are ambiguous

Unambiguous (1 tag)	35,340
Ambiguous (2-7 tags)	4,100
2 tags	3,760
3 tags	264
4 tags	61
5 tags	12
6 tags	2
7 tags	1 (“still”)

POS Tagging: Three Methods

- Rules
- Probabilities (Stochastic)
- Sort of both: Transformation-Based Tagging

Rule-based Tagging

- A two stage architecture
  - Use dictionary (lexicon) to assign each word a list of potential POS
  - Use large lists of hand-written disambiguation rules to identify a single POS for each word.
- ENGTWOL tagger (Voutilainen,’95)
  - 56000 English word stems
- Advantage: high precision (99%)
- Disadvantage: needs a lot of rules

Rules

- Hand-crafted rules for ambiguous words that test the context to make appropriate choices
  - Relies on rules e.g. NP → Det (Adj*) N
    - For example: *the clever student*
  - Morphological Analysis to aid disambiguation
    - E.g. X–ing preceded by Verb – label it a verb
  - 'Supervised method' I.e. using a pre-tagged corpus
    - Advantage: Corpus of same genre
    - Problem: not always available
  - Extra Rules
    - indicative of nouns
    - Punctuation
  - Extremely labor-intensive

Stochastic (Probabilities)

- Simple approach: disambiguate words based on the probability that a word occurs with a particular tag
- N-gram approach: the best tag for given words is determined by the probability that it occurs with the n previous tags
- Viterbi Algorithm: trim the search for the most probable tag using the best N Maximum Likelihood Estimates (n is the number of tags of the following word)
- Hidden Markov Model combines the above two approaches

Stochastic (Probabilities)

- We want the best set of tags for a sequence of words (a sentence)
- W is a sequence of words
- T is a sequence of tags

$$\arg\max P(T \mid W) = \frac{P(W \mid T)P(T)}{P(W)}$$

P(w) is common

Stochastic (Probabilities)

- We want the best set of tags for a sequence of words (a sentence)
- W is a sequence of words
- T is a sequence of tags

$$\arg\max P(T \mid W) = P(W \mid T) P(T)$$

Tag Sequence: P(T)

- How do we get the probability of a specific tag sequence?
  - Count the number of times a sequence occurs and divide by the number of sequences of that length. Not likely.
  - Make a Markov assumption and use N-grams over tags...
    - P(T) is a product of the probability of N-grams that make it up.

P(T): Bigram Example

- <s> Det Adj Adj Noun </s>
- P(Det|<s>)P(Adj|Det)P(Adj|Adj)P(Noun|Adj)

Counts

- Where do you get the N-gram counts?
- From a large hand-tagged corpus.
  - For Bi-grams, count all the Tag_i Tag_{i+1} pairs
  - And smooth them to get rid of the zeroes
- Alternatively, you can learn them from an untagged corpus

What about P(W|T)

- It is asking the probability of seeing "The big red dog" given "Det Adj Adj Noun" !

  - Collect up all the times you see that tag sequence and see how often "The big red dog" shows up. Again not likely to work.

$P(W|T)$

- We'll make the following assumption:
- Each word in the sequence only depends on its corresponding tag. So…

$$P(W \mid T) \approx \prod_{i=1}^{n} P(w_i \mid t_i)$$

- How do we get the statistics for that?

Performance

- This method has achieved 95-96% correct with reasonably complex English tagsets and reasonable amounts of hand-tagged training data.

How accurate are they?

- POS Taggers accuracy rates are in th range of 95-99%
  - Vary according to text/type/genre
    - Of pre-tagged corpus
    - Of text to be tagged
- Worst case scenario: assume success rate of 95%
    - Prob(one-word sentence) = .95
    - Prob(two-word sentence) = .95 * .95 = 90.25%
    - Prob(ten-word sentence) = 59% approx

Thank you

- السلام عليكم ورحمة الله