

Natural Language Processing

By
Prof Grishma S

Text is everywhere!

Social media



Text is everywhere!

Research papers, news, etc.

The screenshot shows a search result for a patent. The title is "Efficiency of quantum vs. classical annealing in nonconvex learning problems". It includes a thumbnail of the patent document, the abstract, and a detailed view of the patent text.

Scientific documents

Patent articles

Books

Review articles

The image is a horizontal collage of logos for various science and technology websites and platforms. From left to right, it includes: 1) The logo for Scientific American, featuring the word 'SCIENTIFIC' in a blue box above 'AMERICAN' in a red box. 2) The ScienceBlogs logo, which consists of a stylized 'S' icon followed by the word 'ScienceBlogs'. 3) The GitHub logo, featuring a black cat icon inside a white circle next to the word 'GitHub' in a large, bold, black sans-serif font. 4) The Stack Overflow logo, featuring a stylized orange flame icon next to the word 'stackoverflow' in a lowercase, bold, black sans-serif font. 5) The ORCID logo, featuring a green circular icon with a white 'iD' monogram inside, followed by the word 'ORCID' in a large, bold, black sans-serif font, with the tagline 'Connecting Research and Researchers' underneath. 6) A section titled 'Crowd-sourced platforms' containing logos for 'Channel Surfing', 'Lifescience', 'Personalized Medicine', 'Epidemiology', and 'Disruptive Biotech'.

03/10/2023

Diversity of Languages (Worldwide)

How many Languages are spoken today?

¹Source: <https://www.ethnologue.com/guides/how-many-languages>

Diversity of Languages (Worldwide)

How many Languages are spoken today?

7,111¹

¹Source: <https://www.ethnologue.com/guides/how-many-languages>

Diversity of Languages (Worldwide)

How many Languages are spoken today?

7,111¹

Can we understand the majority of the

¹Source: <https://www.ethnologue.com/guides/how-many-languages>

Diversity of Languages (Worldwide)

How many Languages are spoken today?

7,111¹

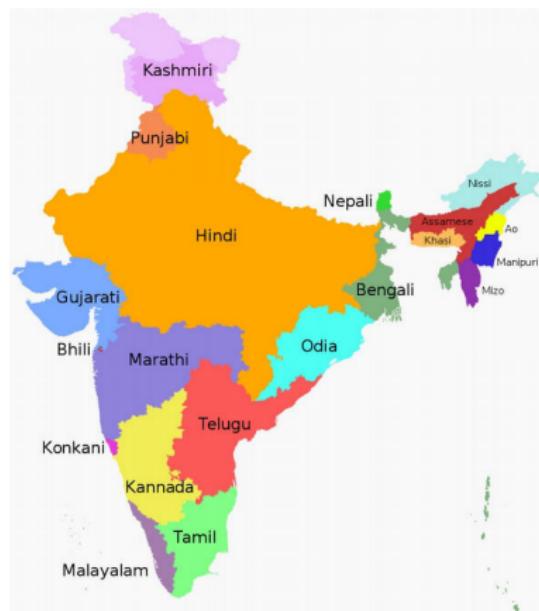
Can we understand the majority of the

Rank ↗	Language	Native speakers in millions 2007 (2010) ↗	Percentage of world population (2007) ↗
1	Mandarin (entire branch)	935 (955)	14.1%
2	Spanish	390 (405)	5.85%
3	English	365 (360)	5.52%
4	Hindi ^[b]	295 (310)	4.46%
5	Arabic	280 (295)	4.23%
6	Portuguese	205 (215)	3.08%
7	Bengali (Bangla)	200 (205)	3.05%
8	Russian	160 (155)	2.42%
9	Japanese	125 (125)	1.92%
10	Punjabi	95 (100)	1.44%

¹Source: <https://www.ethnologue.com/guides/how-many-languages>

Diversity of Languages (India)

Can we understand the majority of the India's data?



Total Languages: > 1650 (2011 consensus)

Hindi: 57.1%

English: 10.6%

Bengali: 8.9%

Marathi: 8.2%

Telugu: 7.8 %

...

...

The goals of NLP

French Sentence: Tu Bois un Coca-cola

The goals of NLP

French Sentence: Tu Bois un Coca-cola

English Translation: You drink a Coca-cola

The goals of NLP

French Sentence: Tu Bois un Coca-cola

English Translation: You drink a Coca-cola

We try to understand a foreign language using some known keywords

The goals of NLP

French Sentence: Tu Bois un Coca-cola

English Translation: You drink a Coca-cola

We try to understand a foreign language using some known keywords

Goals of NLP

- Deep understanding of broad language constructs.
- Achieve human-like comprehension of texts/languages.
- Make computer systems to understand, draw inferences from, summarize, translate and generate accurate and natural human text and language.

Some Applications: Language Translation

The screenshot shows the Google Translate interface. At the top, there's a navigation bar with the Google logo, followed by "Translate" and a "Turn off instant translation" link. Below the navigation bar, there are language selection buttons for English, Spanish, Hindi, and Detect language. The main area shows a translation from English to Hindi. The English input is "Let's go out for a date". The Hindi output is "चलो एक तारीख के लिए बाहर जाओ". Below the translation, there are edit controls (undo, redo, etc.) and a "Suggest an edit" link. The bottom of the interface shows the original text in Hindi: "chalo ek taareekh ke lie baahar jao".

Language Translation

Google Translate

Turn off instant translation

English Spanish Hindi Detect language English Spanish Hindi Translate

मेरे हाथ में तेरा हाथ हो सारी जन्नतें मेरे साथ हो। All hands should be with me in my hand.

49/5000 Suggest an edit

mere haath mein tera haath ho saaree jannaten mere saath ho

Language Translation is not easy even for humans

Pepsi Chinese blunder

“Come alive with the Pepsi Generation”, when translated into Chinese meant,
“Pepsi brings your relatives back from the dead.”

Language Translation is not easy even for humans

Pepsi Chinese blunder

“Come alive with the Pepsi Generation”, when translated into Chinese meant, “Pepsi brings your relatives back from the dead.”

KFC's Chinese blunder

KFC's slogan, “Finger lickin' good”, when translated into Chinese meant “We'll eat your fingers off.”

Some more examples...



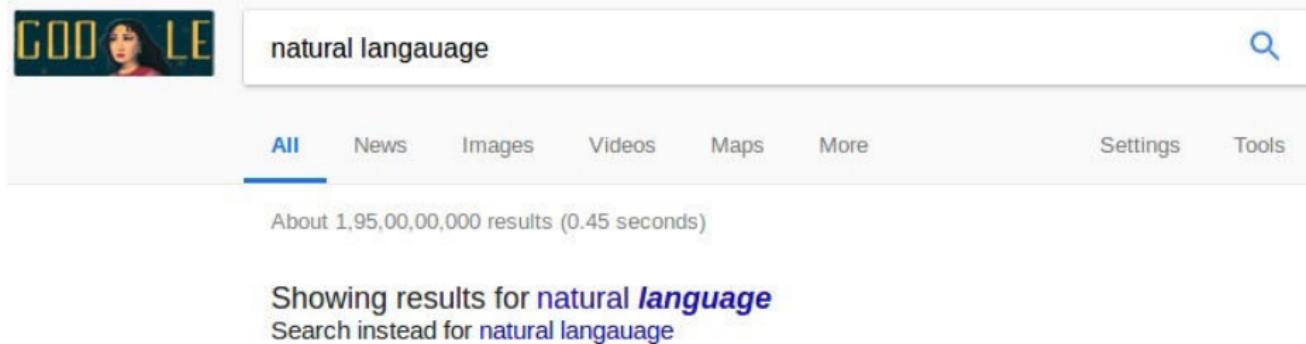
Query Recommendation in Search Engines



Dhoni is

- dhoni is **from which state**
- dhoni is **back**
- dhoni is **in which team**
- dhoni is
- dhoni is **retiring**
- dhoni is **best**
- dhoni is **photo**
- dhoni is **a legend**
- dhoni is **injured**
- dhoni is **age**

Spelling Correction



A screenshot of a Google search results page. The search bar at the top contains the query "natural langauge". Below the search bar, the "All" tab is selected, followed by "News", "Images", "Videos", "Maps", and "More". To the right of these tabs are "Settings" and "Tools". A blue horizontal bar spans across the page. Below the tabs, it says "About 1,95,00,00,000 results (0.45 seconds)". The main content area displays a snippet: "Showing results for natural *language*" followed by "Search instead for natural langauge".

Information Extraction

New York Times Co. named Russell T. Lewis, 45, president and general manager of its flagship New York Times newspaper, responsible for all business-side activities. He was executive vice president and deputy general manager. He succeeds Lance R. Primis, who in September was named president and chief operating officer of the parent.

Person	Company	Post	State
Russell T. Lewis	New York Times newspaper	president and general manager	start
Russell T. Lewis	New York Times newspaper	executive vice president	end
Lance R. Primis	New York Times Co.	president and CEO	start

Sentiment Analysis



Saravana Kumar



Worst software update Needed

4 May 2019

Style: 3GB RAM | Size: 32GB | Colour: Elegant Blue | **Verified Purchase**

Bad software

Phone hangs a lot

Many issues and bugs in phone

Only Battery and front camera is good

Worst from MI

208 people found this helpful

Helpful

▼ 1 Comment

| Report abuse

Recent Trends: Fake news detection

Lies

IndiaToday.in
New Delhi, April 13, 2017 | UPDATED 18:31 IST

Uttar Pradesh government ends reservation for SC, ST, OBC candidates in private medical colleges

The Uttar Pradesh government made an announcement to end caste-based reservations in private medical and dental colleges of up. An order has been passed to do away with the quota for candidates belonging to SC, ST and OBC categories.



Uttar Pradesh CM Yogi Adityanath

RELATED STORIES

- ❑ Union Cabinet approves IIPV Visakhapatnam's status as institute of national importance
- ❑ Delhi HC suggests outsiders should be banned from entering JNU
- ❑ 25,000 Indian students to be trained in cyber security by Cisco
- ❑ Haryana boy makes it to IIM-A with 89.67 percentile

Truth

'Caste-based reservation never existed in private medical colleges in Uttar Pradesh'

Shalivaa Sharda THN | Apr 13, 2017, 07:27 PM IST

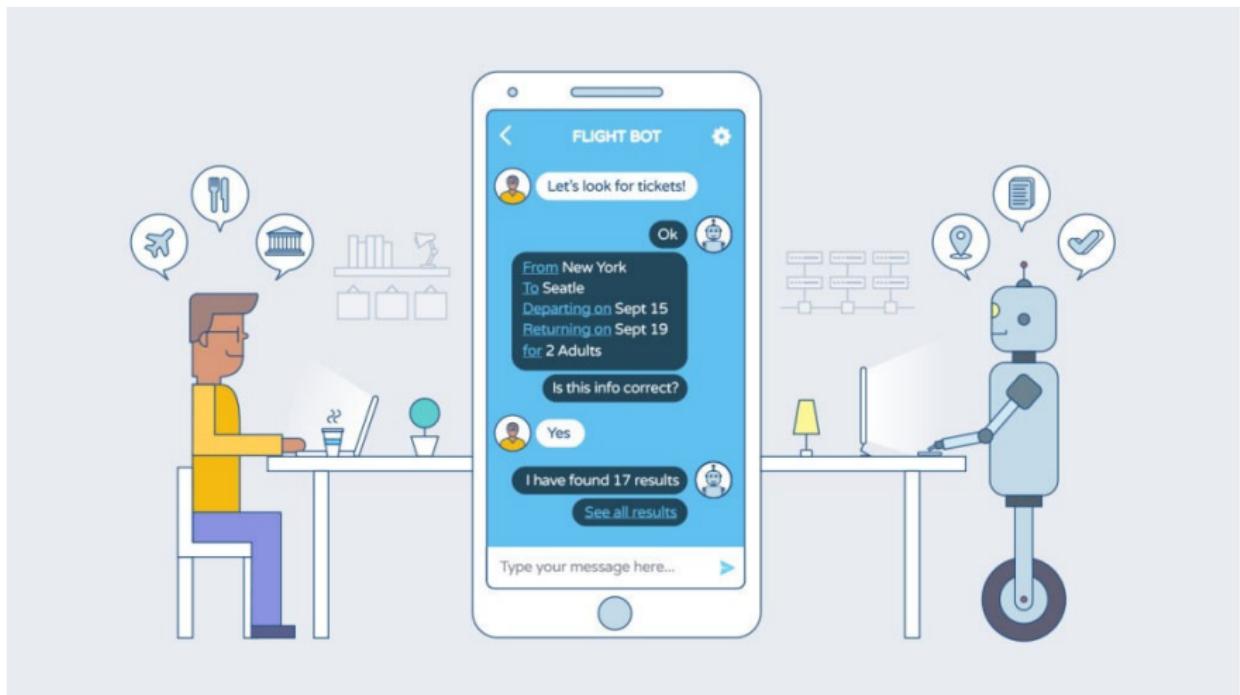


Pile photograph used for representational purpose

HIGHLIGHTS

- ❑ UP medical education department refuted reports of Yogi government abolishing reservations in private medical colleges
- ❑ It said caste-based reservation was never a part of the admission process in private colleges in UP

Recent Trends: Chatbots



Other Goals

- Text Summarization
- Opinion dynamics
- Spam detection
- . . .

Other Goals

- Text Summarization
- Opinion dynamics
- Spam detection
- . . .

Natural Language
Technology not yet
perfect

But still good enough for
several useful
applications

Why is NLP hard?

Compounding

निरन्तरान्धकारित-दिगन्तर-कन्दलदमन्द-सुधारस-विन्दु-सान्दरतर-घनाघन-वृन्द-
सन्देहकर-स्यन्दमान-मकरन्द-विन्दु-बन्धुरतर-माकन्द-तरु-कुल-तल्प-कल्प-
मृदुल-सिकता-जाल-जटिल-मूल-तल-मरुवक-मिलदलधु-लघु-लय-कलित-
रमणीय-पानीय-शालिका-बालिका-करार-विन्द-गलन्तिका-गलदेला-लवड़ग-
पाटल-घनसार-कस्तूरिकातिसौरभ-मेदुर-लघुतर-मधुर-शीतलतर-सलिलधारा-
निराकरिष्णु-तदीय-विमल-विलोचन-मयूख-रेखापसारित-पिपासायास-पथिक-
लोकान्

195 characters (with 428 characters when transliterated into the roman writing system).

Why is NLP hard?

Ambiguity

Lexical Ambiguity

The presence of two or more possible meanings within a single word.



"I saw her duck."

Syntactic Ambiguity

The presence of two or more possible meanings within a single sentence or sequence of words.



"The chicken is ready to eat."

Why is NLP hard?

Ambiguity



Why else is NLP hard?

Shorthand text



Why else is NLP hard?

Non-standard English

Why else is NLP hard?

Segmentation Issues

the New York New Haven Railroad

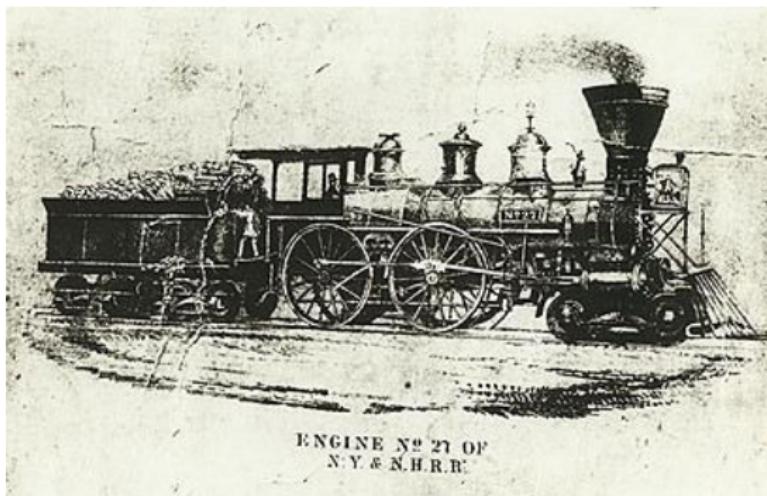
Why else is NLP hard?

Segmentation Issues

the New York New Haven Railroad

the [New] [York New] [Haven] [Railroad]

the [New York] [New Haven] [Railroad]



Why else is NLP hard?

Idioms

- Dark horse
- Ball in your court
- Burn the midnight oil

Why else is NLP hard?

Idioms : An expression whose meaning is different from the meanings of the individual words in it.

Idioms Example

- Dark horse
- Ball in your court Burn the midnight oil
-

Neologisms: A new word, phrase or expression, or a new meaning of a familiar word

- Unfriend
- Retweet
- Google/Skype/photoshop

Why is NLP hard?

New Senses of a word

- That's *sick* dude!
- Giants

Why is NLP hard?

New Senses of a word

- That's *sick* dude!

Giants ...

*multinationals,
conglomerates,
manufacturers*

Why is NLP hard?

New Senses of a word

- That's *sick* dude!

Giants ...

*multinationals,
conglomerates,
manufacturers*



Tricky Entity Names

Where is *A Bug's Life*
playing ...

Let It Be was recorded

...

Why is NLP hard?

Code Mixing/switching



Romanization

BAHUT TENSION HAI O
PUKKA IDIOT!
EK CHANCE MILEGA?
CUTTING CHAI
ADJUST KIJIYE
CHUTNEFYING
ENGLISH

What we do in NLP?

Create annotated corpora

Brown Corpus, Google Books Ngram Corpus, Reuters Newswire Topic Classification, IMDB Movie Review Sentiment Classification, Project Gutenberg, etc.

Create Models/Algorithms

LDA, BERT, CKY, Edit Distance, CRF++, etc.

Create Tools

CoreNLP, NLTK, Gensim, SpaCy, etc.

Stages in NLP (*traditional view*)

- *Phonetics and phonology*
- *Morphology*
- *Lexical Analysis*
- *Syntactic Analysis*
- *Semantic Analysis*
- *Pragmatics*
- *Discourse*

Source: IITB NLP Course by Pushpak Bhattacharyya

Phonetics

- Processing of speech
- Challenges
 - Homophones: *bank (finance)* vs. *bank (river bank)*
 - Near Homophones: *maatraqa* vs. *maatra (hin)*
 - Word Boundary
 - *aajaayenge (aa jaayenge (will come) or aaj aayenge (will come today)*
 - *I got [ua]plate*
 - Phrase boundary
 - *mtech1 students are especially exhorted to attend as such seminars are integral to one's post-graduate education*
 - Disfluency: *ah, um, ahem etc.*

Morphology

- Word formation rules from *root words*
- Nouns: Plural (*boy-boys*); Gender marking (*czar-czarina*)
- Verbs: Tense (*stretch-stretched*); Aspect (e.g. perfective *sit-had sat*); Modality (e.g. request *khaanaa* \sqcup *khaaiie*)
- First crucial first step in NLP
- Languages rich in morphology: e.g., Dravidian, Hungarian, Turkish
- Languages poor in morphology: Chinese, English
- Languages with rich morphology have the advantage of easier processing at higher stages of processing
- A task of interest to computer science: *Finite State Machines for Word Morphology*

Lexical Analysis

- Essentially refers to dictionary access and obtaining the properties of the word
 - e.g. *dog*
 - noun (lexical property)*
 - take-'s'-in-plural (morph property)*
 - animate (semantic property)*
 - 4-legged (-do-)*
 - carnivore (-do)*
- Challenge: *Lexical or word sense disambiguation*

Lexical Disambiguation

First step: *part of Speech Disambiguation*

Dog as a noun (animal)

Dog as a verb (*to pursue*)

Sense Disambiguation

Dog (as *animal*)

Dog (as a very detestable person)

Needs word relationships in a context

The chair emphasized the need for adult education

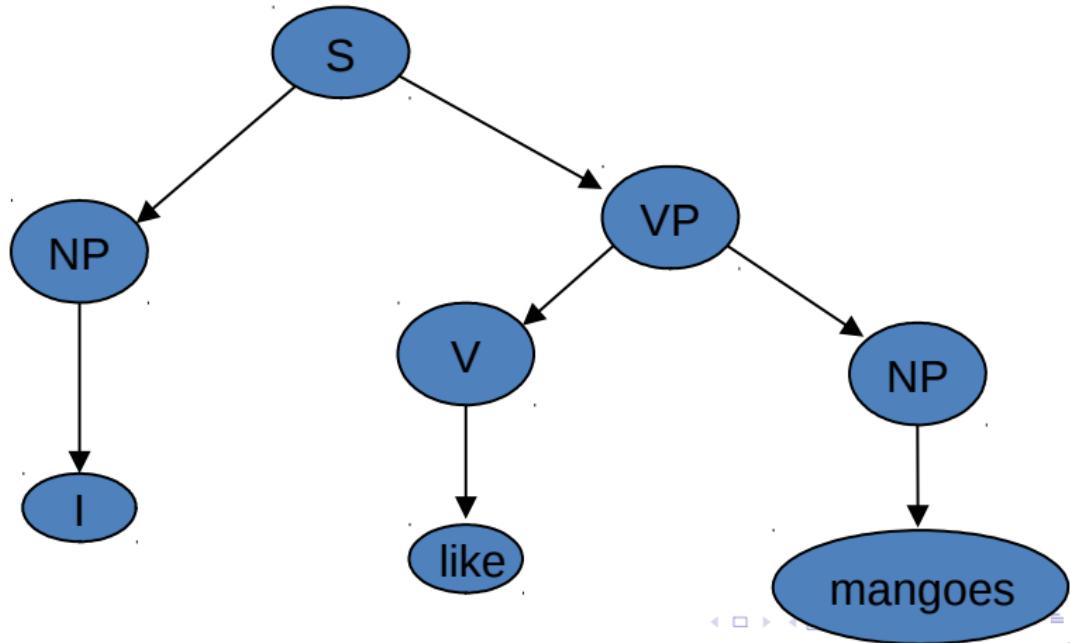
Very common in day to day communications

Satellite Channel Ad: *Watch what you want, when you want* (two senses of watch)

e.g., Ground breaking ceremony/research

Syntax Processing Stage

Structure Detection



Parsing Strategy

Driven by grammar

$S \rightarrow NP\ VP$

$NP \rightarrow N \mid PRON$

$VP \rightarrow V\ NP \mid V\ PP$

$N \rightarrow \text{Mangoes}$

$PRON \rightarrow I$

$V \rightarrow \text{like}$

Challenges in Syntactic Processing: Structural Ambiguity

- *Scope Ambiguity*

1. *The old men and women were taken to safe locations*
(old men and women) vs. ((old men) and women)
2. *No smoking areas will allow Hookas inside*

- *Preposition Phrase Attachment*

- *I saw the boy with a telescope*
(who has the telescope?)
- I saw the mountain with a telescope
(world knowledge: *mountain* cannot be an *instrument of seeing*)
- I saw the boy with the pony-tail
(world knowledge: *pony-tail* cannot be an *instrument of seeing*)
- Very ubiquitous: newspaper headline “*20 years later, BMC pays father 20 lakhs for causing son’s death*”

Structural Ambiguity...

Overheard

I did not know my PDA had a phone for 3 months

An actual sentence in the newspaper

The camera man shot the man with the gun when he was near Tendulkar

Semantic Analysis

- *Representation in terms of*
Predicate calculus/Semantic
Nets/Frames/Conceptual Dependencies
and Scripts
- *John gave a book to Mary*
Give action: Agent: John, Object: Book,
Recipient: Mary
- *Challenge: ambiguity in semantic role labeling*
(Eng) Visiting aunts can be a nuisance
(Hin) aapko mujhe mithaai khilaanii padegii
(ambiguous in Marathi and Bengali too; not
in Dravidian languages)

Pragmatics

- *Very hard problem*
- *Model user intention*
 - *Tourist (in a hurry, checking out of the hotel, motioning to the service boy): Boy, go upstairs and see if my sandals are under the divan. Do not be late. I just have 15 minutes to catch the train.*
 - *Boy (running upstairs and coming back panting): yes sir, they are there.*
- *World knowledge*
 - *WHY INDIA NEEDS A SECOND OCTOBER*

Discourse

- Processing of sequence of sentences
 - *Mother to John:*
 - *John go to school. It is open today. Should you bunk? Father will be very angry.*
- Ambiguity of *open*
- *bunk what?*
- *Why will the father be angry?*
 - *Complex chain of reasoning and application of world knowledge*
 - *Ambiguity of father*
father as parent
or
father as headmaster

Reference Books

- Daniel Jurafsky and James H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Speech Recognition, and Computational Linguistics*. 2nd edition. Prentice-Hall. 2009.
- Christopher D. Manning and Hinrich Schütze. *Foundations of Statistical Natural Language Processing*. MIT Press. 1999.
- Steven Bird, Ewan Klein and Edward Loper. *Natural Language Processing with Python*. O'Reilly Media. 2009.
- Ian Goodfellow, Yoshua Bengio and Aaron Courville. *Deep Learning*. MIT Press. 2016.

Regular Expressions and Finite State Automata

Mausam

(Based on slides by Jurafsky & Martin,
Julia Hirschberg)

Regular Expressions and Text Searching

- Everybody does it
 - ◆ Emacs, vi, perl, grep, etc..
- Regular expressions are a compact textual representation of a set of strings representing a language.

RE	Example Patterns Matched
/woodchucks/	“interesting links to <u>woodchucks</u> and lemurs”
/a/	“ <u>Mary</u> Ann stopped by <u>Mona</u> ’s”
/Claire_says,/	“ “Dagmar, my gift please,” <u>Claire</u> says,”
/DOROTHY/	“SURRENDER <u>DOROTHY</u> ”
/ ! /	“You’ve left the burglar behind again!” said Nori

Regular Expressions

RE	Match	Example Patterns
/ [wW]oodchuck /	Woodchuck or woodchuck	“ <u>Woodchuck</u> ”
/ [abc] /	‘a’, ‘b’, <i>or</i> ‘c’	“In uomini, in soldati”
/ [1234567890] /	any digit	“plenty of <u>7</u> to <u>5</u> ”

Regular Expressions

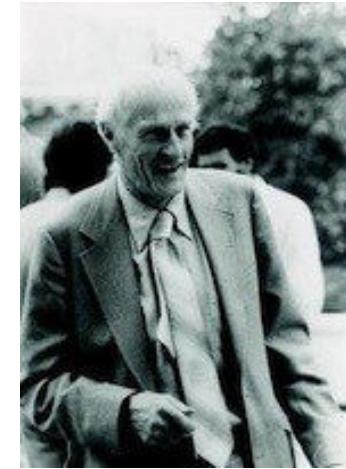
RE	Match	Example Patterns Matched
/ [A-Z] /	an upper case letter	“we should call it ‘ <u>Drenched Blossoms</u> ’ ”
/ [a-z] /	a lower case letter	“ <u>my</u> beans were impatient to be hoed!”
/ [0-9] /	a single digit	“Chapter <u>1</u> : Down the Rabbit Hole”

Regular Expressions

RE	Match (single characters)	Example Patterns Matched
[^A-Z]	not an upper case letter	“Oyfn pripetchik”
[^Ss]	neither ‘S’ nor ‘s’	“I have no exquisite reason for’t”
[^\.\]	not a period	“our resident Djinn”
[e^]	either ‘e’ or ‘^’	“look up ^ now”
a^b	the pattern ‘a^b’	“look up a^ b now”

Regular Expressions: ? * + .

Pattern	Matches	
colou?r	Optional previous char	<u>color</u> <u>colour</u>
oo*h!	0 or more of previous char	<u>oh!</u> <u>ooh!</u> <u>oooh!</u> <u>ooooh!</u>
o+h!	1 or more of previous char	<u>oh!</u> <u>ooh!</u> <u>oooh!</u> <u>ooooh!</u>
baa+		<u>baa</u> <u>baaa</u> <u>baaaa</u> <u>baaaaa</u>
beg.n		<u>begin</u> <u>begun</u> <u>begun</u> <u>beg3n</u>



Stephen C Kleene

Kleene *, Kleene -

Regular Expressions: Anchors

^ \$

Pattern	Matches
^ [A-Z]	Palo Alto
^ [^A-Za-z]	<u>1</u> "Hello"
\. \$	The end <u>.</u>
. \$	The end <u>?</u> The end <u>!</u>

Regular Expressions

RE	Expansion	Match	Examples
\d	[0-9]	any digit	Party_of_5
\D	[^0-9]	any non-digit	Blue_moon
\w	[a-zA-Z0-9_]	any alphanumeric/underscore	Daiyu
\W	[^\w]	a non-alphanumeric	!!!
\s	[\r\t\n\f]	whitespace (space, tab)	
\S	[^\s]	Non-whitespace	in_Concord

Regular Expressions

RE	Match	Example Patterns Matched
*	an asterisk “*”	“K*A*P*L*A*N”
\.	a period “.”	“Dr. <u>_</u> Livingston, I presume”
\?	a question mark	“Why don’t they come and lend a hand <u>_</u> ?”
\n	a newline	
\t	a tab	

Example

- Find all the instances of the word “the” in a text.
 - ◆ `/the/`
 - ◆ `/ [tT] he/`
 - ◆ `/\b [tT] he \b/`
 - ◆ `[^a-zA-Z] [tT] he [^a-zA-Z]`
 - ◆ `(^ | [^a-zA-Z]) [tT] he ($ | [^a-zA-Z])`

Errors

- The process we just went through was based on two fixing kinds of errors
 - ◆ Matching strings that we should not have matched (**there**, **then**, **other**)
 - **False positives** (Type I)
 - ◆ Not matching things that we should have matched (**The**)
 - **False negatives** (Type II)

Errors

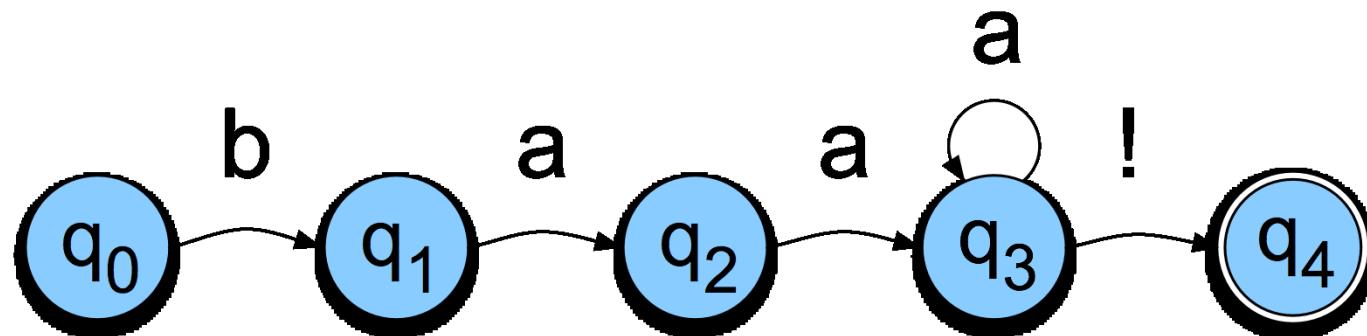
- We'll be telling the same story for many tasks, all semester. Reducing the error rate for an application often involves two **antagonistic** efforts:
 - ◆ Increasing accuracy, or precision, (minimizing false positives)
 - ◆ Increasing coverage, or recall, (minimizing false negatives).

Finite State Automata

- Regular expressions can be viewed as a textual way of specifying the structure of finite-state automata.
- FSAs capture significant aspects of what linguists say we need for **morphology** and parts of **syntax**.

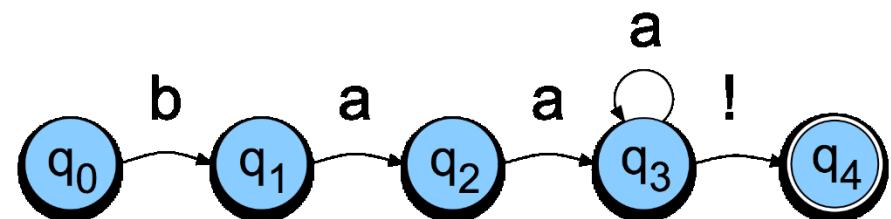
FSAs as Graphs

- Let's start with the sheep language from Chapter 2
 - /baa+ ! /



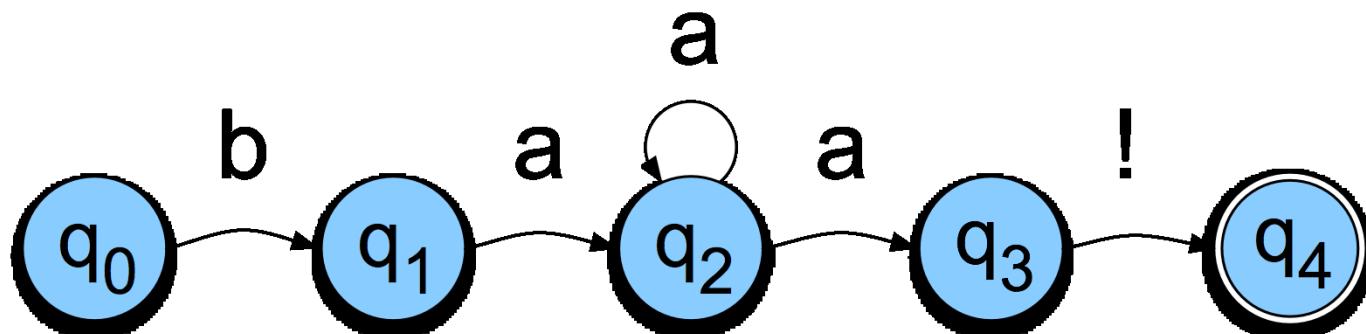
Sheep FSA

- We can say the following things about this machine
 - ◆ It has 5 states
 - ◆ **b**, **a**, and **!** are in its alphabet
 - ◆ q_0 is the start state
 - ◆ q_4 is an accept state
 - ◆ It has 5 transitions



But Note

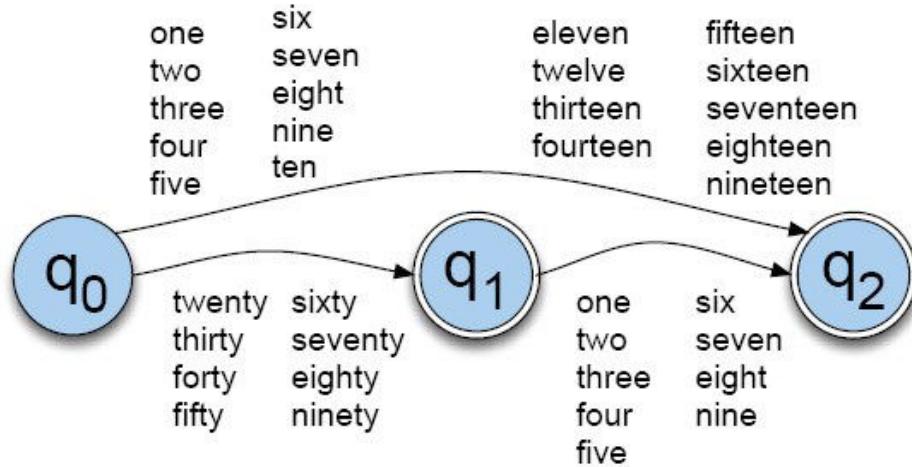
- There are other machines that correspond to this same language



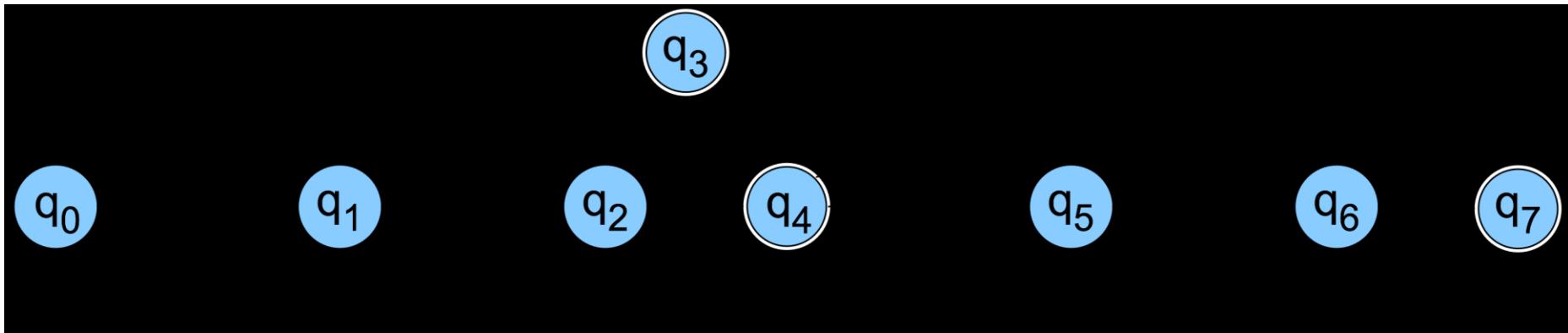
More Formally

- You can specify an FSA by enumerating the following things.
 - ◆ The set of states: Q
 - ◆ A finite alphabet: Σ
 - ◆ A start state
 - ◆ A set of accept/final states
 - ◆ A transition function that maps $Q \times \Sigma$ to Q

Dollars and Cents



Dollars and Cents

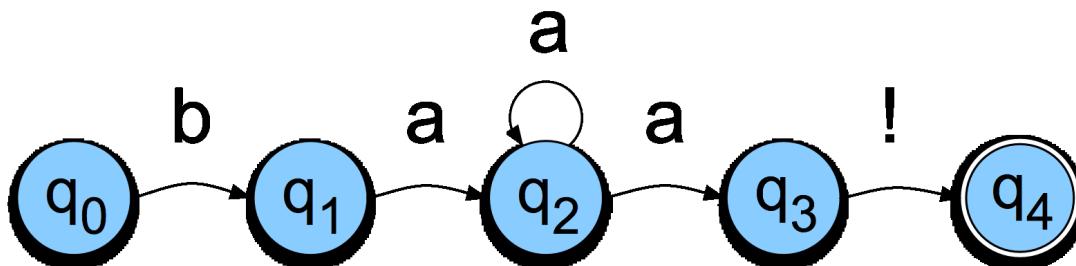


Yet Another View

- The guts of FSAs can ultimately be represented as tables

If you're in state 1 and you're looking at an a, go to state 2

	b	a	!	e
0		1		
1			2	
2			2,3	
3				4
4				

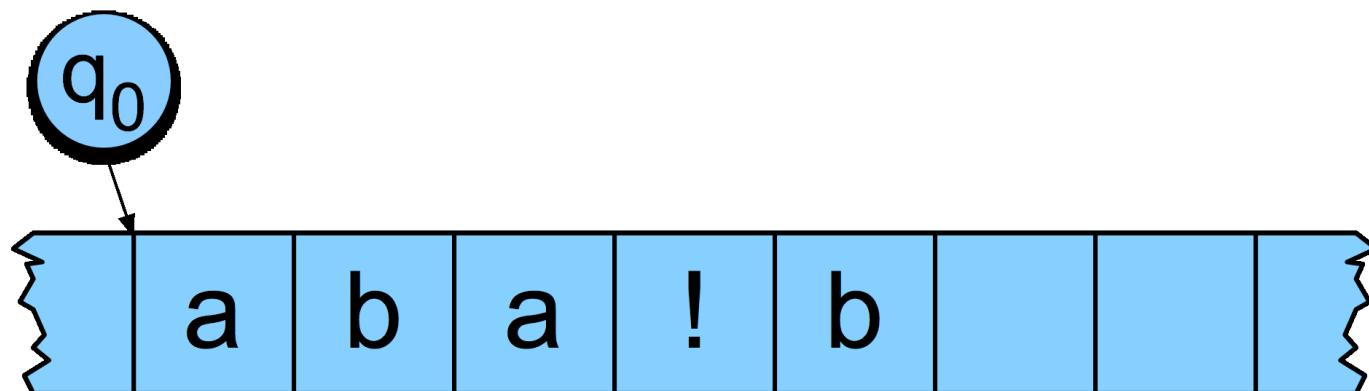


Recognition

- Recognition is the process of determining if a string should be accepted by a machine
- Or... it's the process of determining if a string is in the language we're defining with the machine
- Or... it's the process of determining if a regular expression matches a string
- Those all amount the same thing in the end

Recognition

- Traditionally, (Turing's notion) this process is depicted with a tape.



Recognition

- Simply a process of starting in the start state
- Examining the current input
- Consulting the table
- Going to a new state and updating the tape pointer.
- Until you run out of tape.

D-Recognize

function D-RECOGNIZE(*tape, machine*) **returns** accept or reject

index \leftarrow Beginning of tape

current-state \leftarrow Initial state of machine

loop

if End of input has been reached **then**

if current-state is an accept state **then**

return accept

else

return reject

elsif *transition-table[current-state,tape[index]]* is empty **then**

return reject

else

current-state \leftarrow *transition-table[current-state,tape[index]]*

index \leftarrow *index* + 1

end

Key Points

- Deterministic means that at each point in processing there is always one unique thing to do (no choices).
- D-recognize is a simple table-driven interpreter
- The algorithm is universal for all unambiguous regular languages.
 - ◆ To change the machine, you simply change the table.

Key Points

- Crudely therefore... matching strings with regular expressions (ala Perl, grep, etc.) is a matter of
 - ◆ translating the regular expression into a machine (a table) and
 - ◆ passing the table and the string to an interpreter

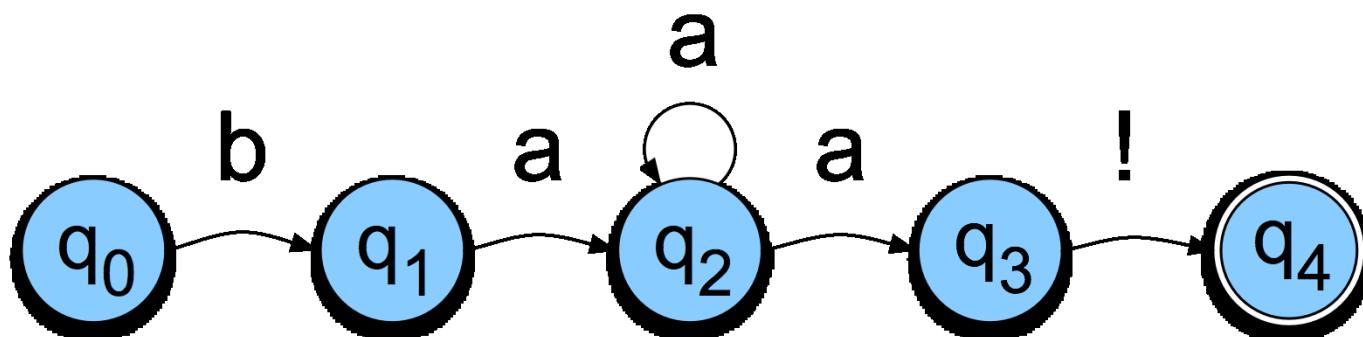
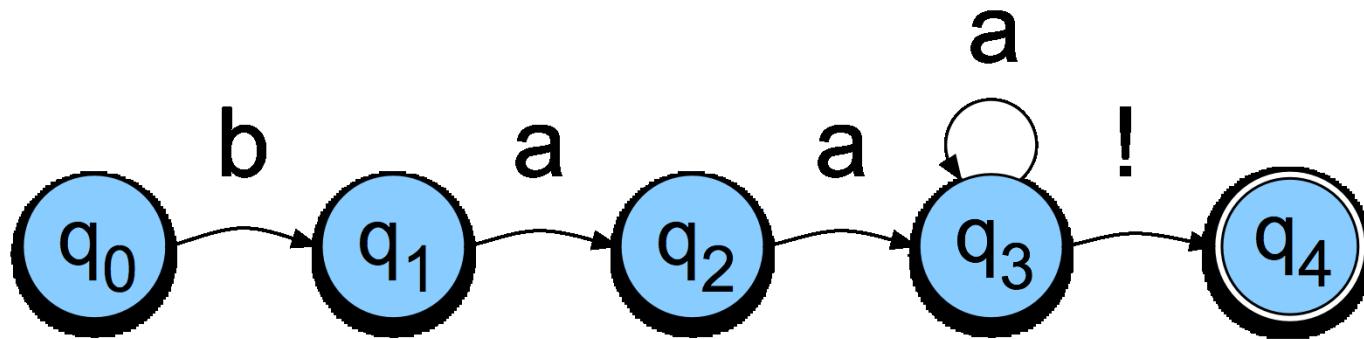
Generative Formalisms

- *Formal Languages* are sets of strings composed of symbols from a finite set of symbols.
- Finite-state automata define formal languages (without having to enumerate all the strings in the language)
- The term *Generative* is based on the view that you can run the machine as a generator to get strings from the language.

Generative Formalisms

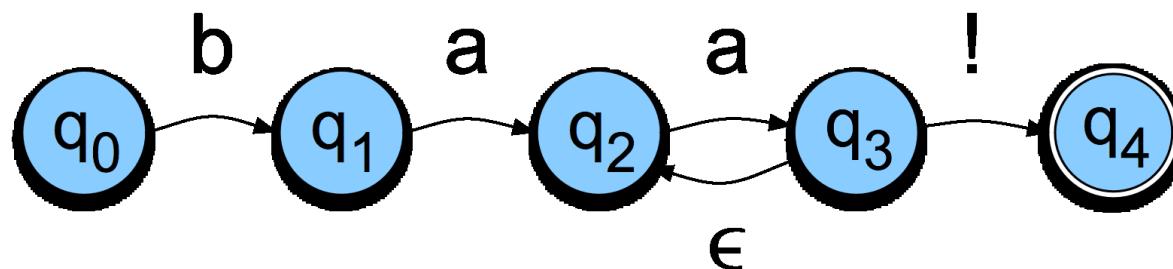
- FSAs can be viewed from two perspectives:
 - ◆ Acceptors that can tell you if a string is in the language
 - ◆ Generators to produce *all and only* the strings in the language

Non-Determinism



Non-Determinism cont.

- Yet another technique
 - ◆ Epsilon transitions
 - ◆ Key point: these transitions do not examine or advance the tape during recognition



Equivalence

- Non-deterministic machines can be converted to deterministic ones with a fairly simple construction
- That means that they have the same power; non-deterministic machines are not more powerful than deterministic ones in terms of the languages they can accept

ND Recognition

- Two basic approaches (used in all major implementations of regular expressions, see Friedl 2006)
 1. Either take a ND machine and convert it to a D machine and then do recognition with that.
 2. Or explicitly manage the process of recognition as a state-space search (leaving the machine as is).

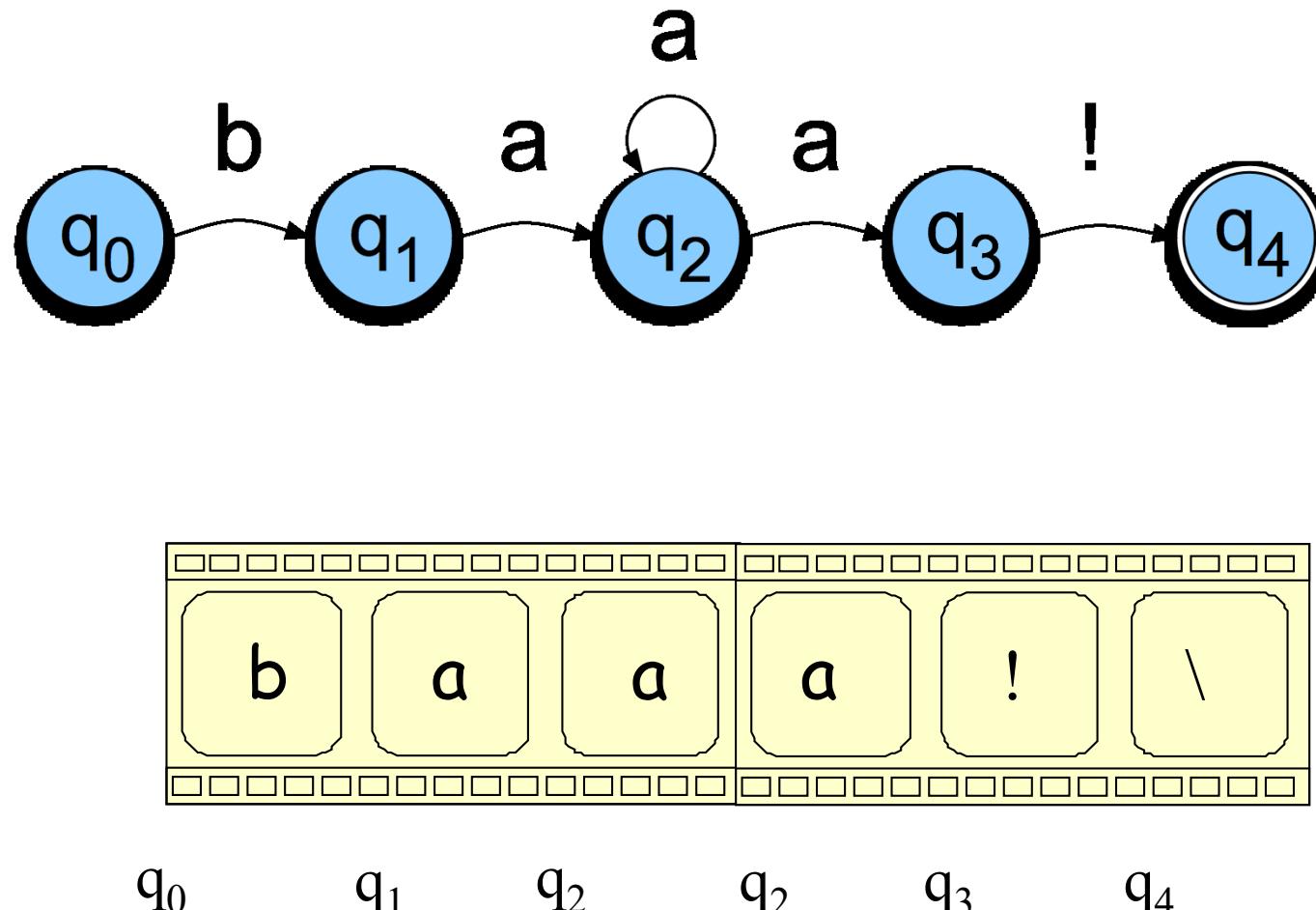
Non-Deterministic Recognition: Search

- In a ND FSA **there exists** at least one path through the machine for a string that is in the language defined by the machine.
- **But not all paths** directed through the machine for an accept string lead to an accept state.
- **No paths** through the machine lead to an accept state for a string not in the language.

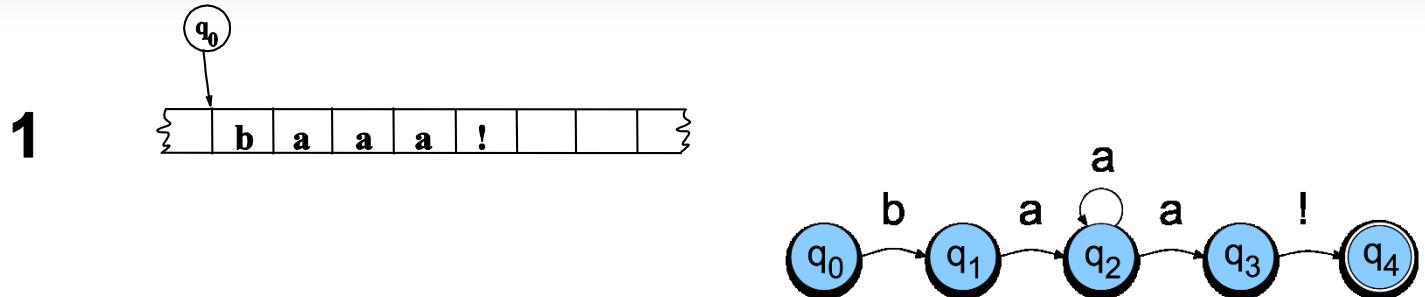
Non-Deterministic Recognition

- So **success** in non-deterministic recognition occurs when a path is found through the machine that ends in an accept.
- **Failure** occurs when **all** of the possible paths for a given string lead to failure.

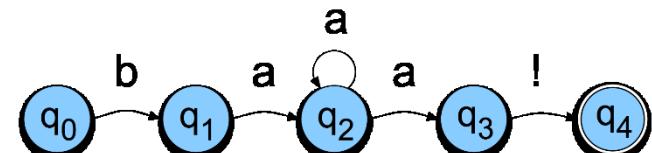
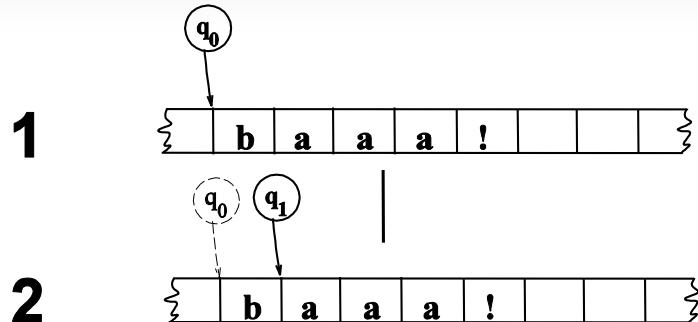
Example



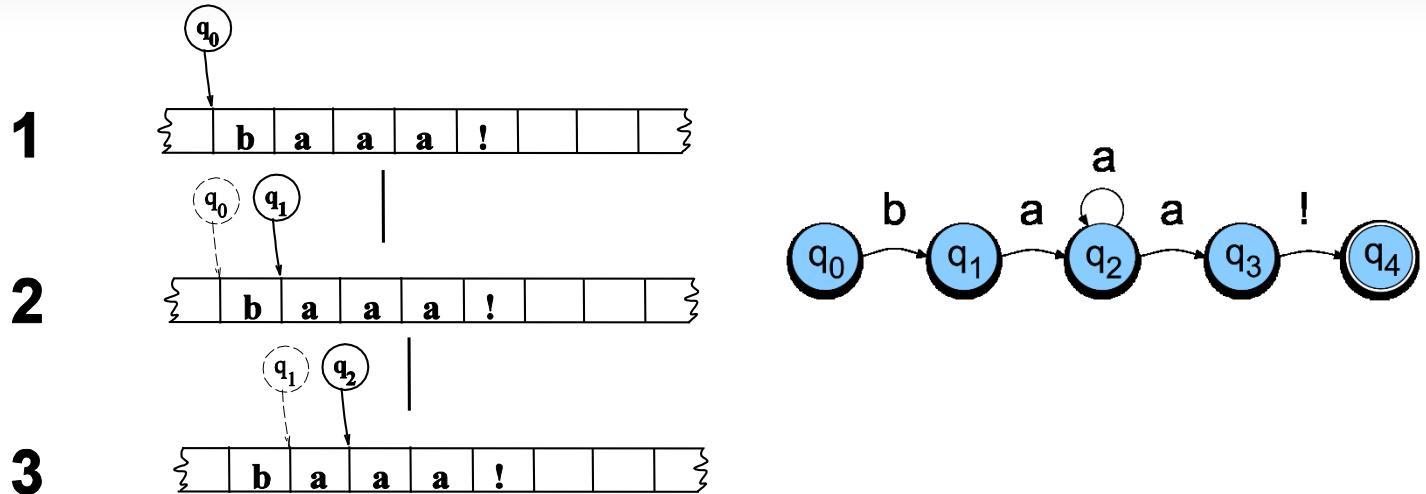
Example



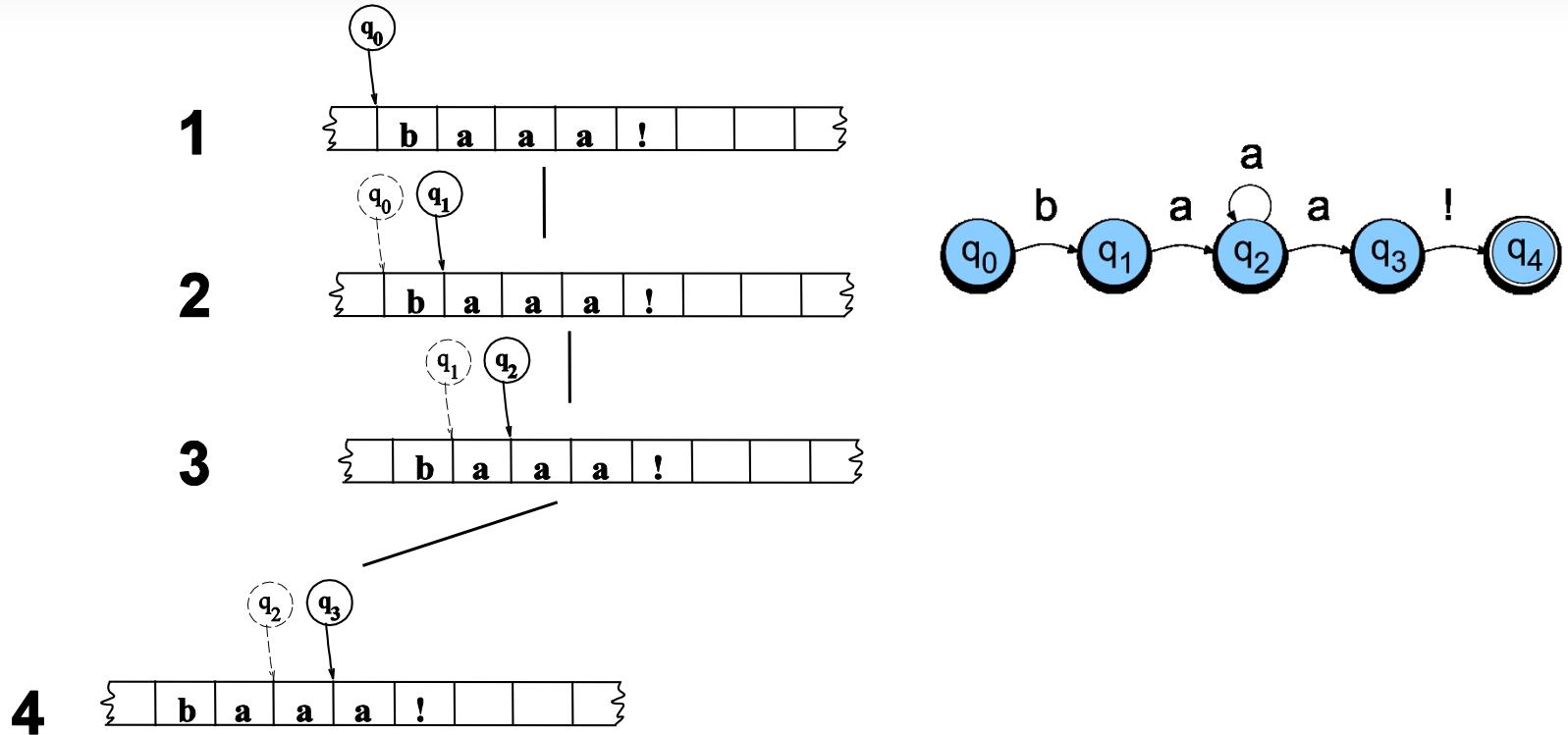
Example



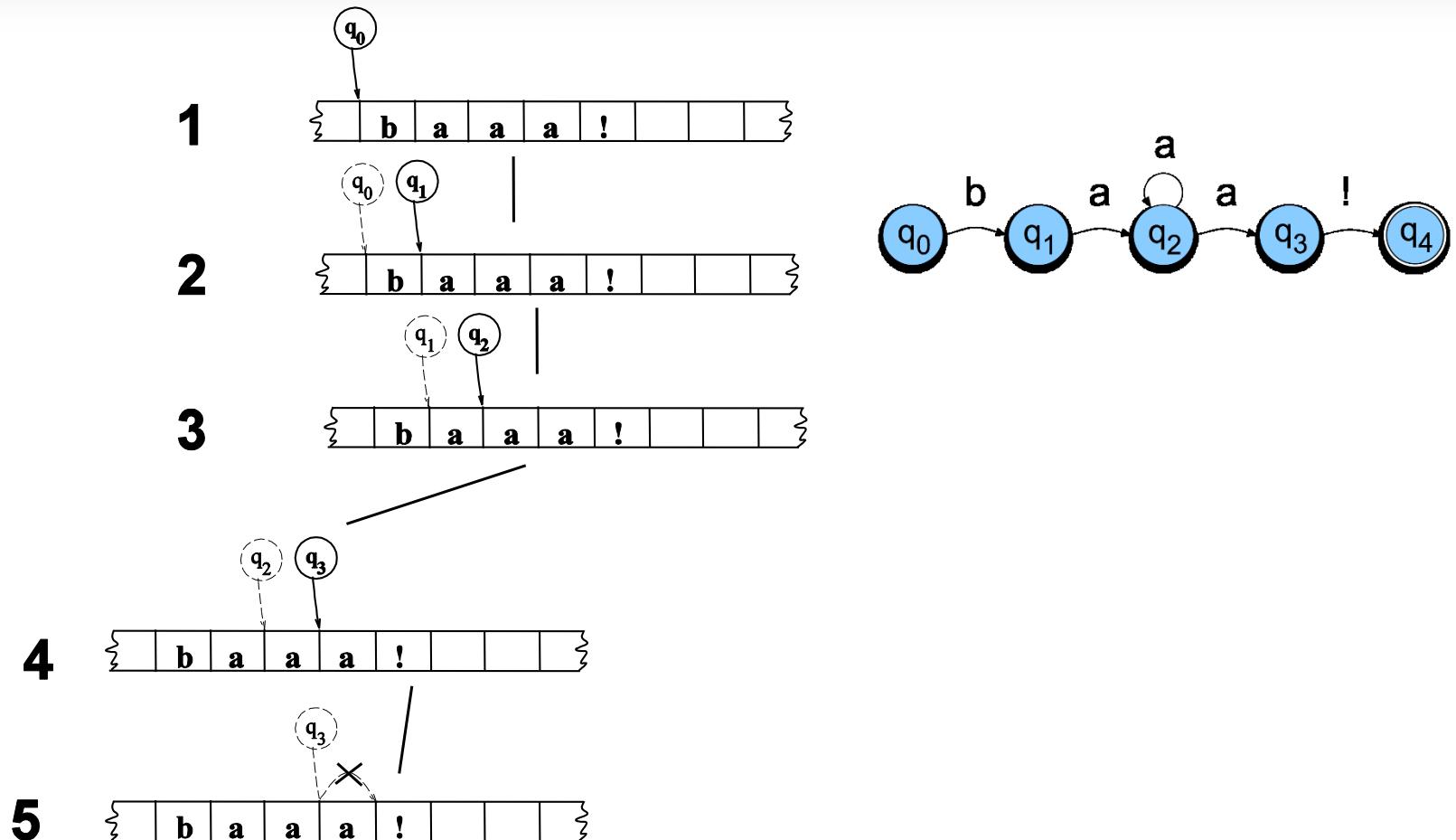
Example



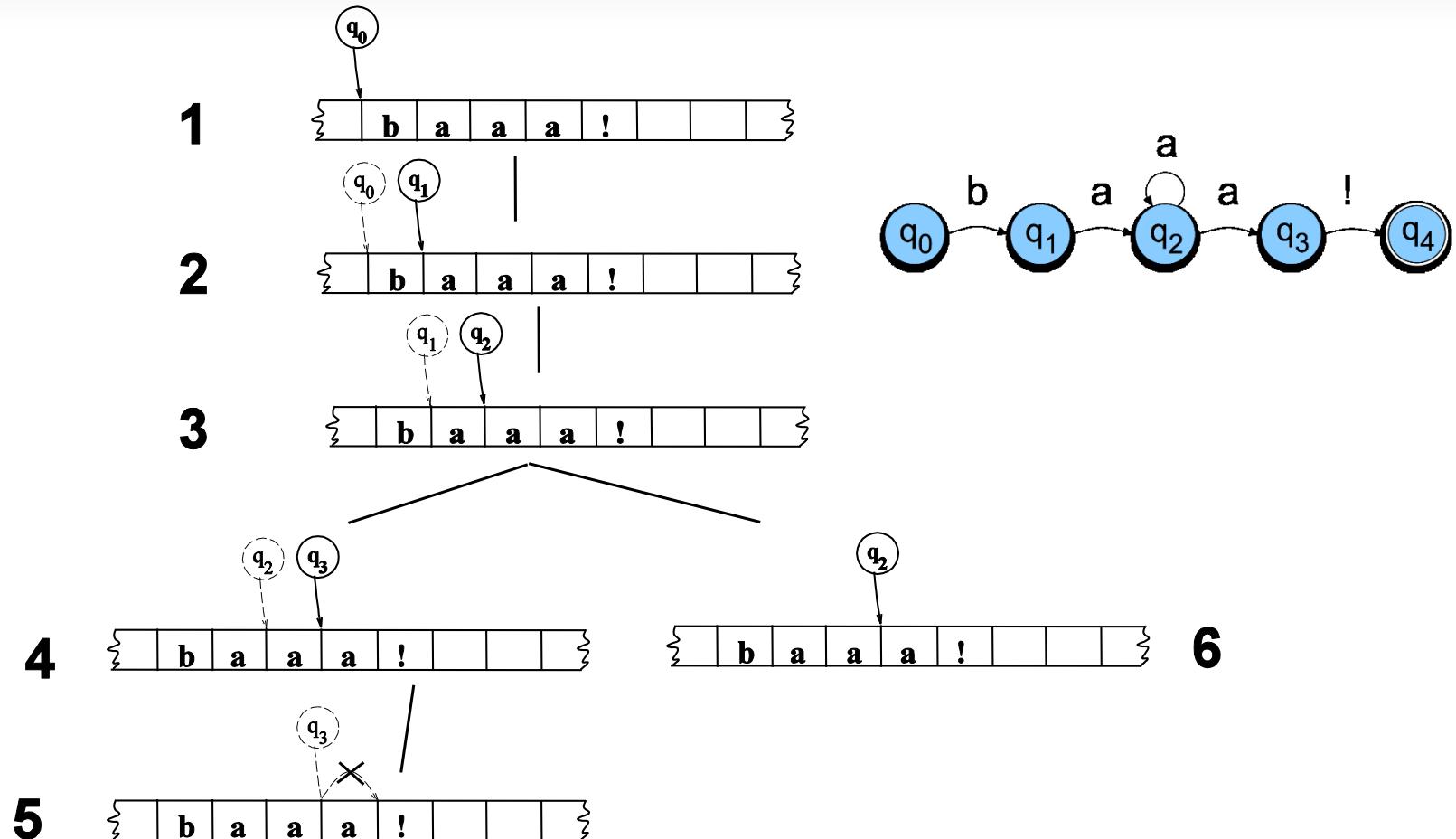
Example



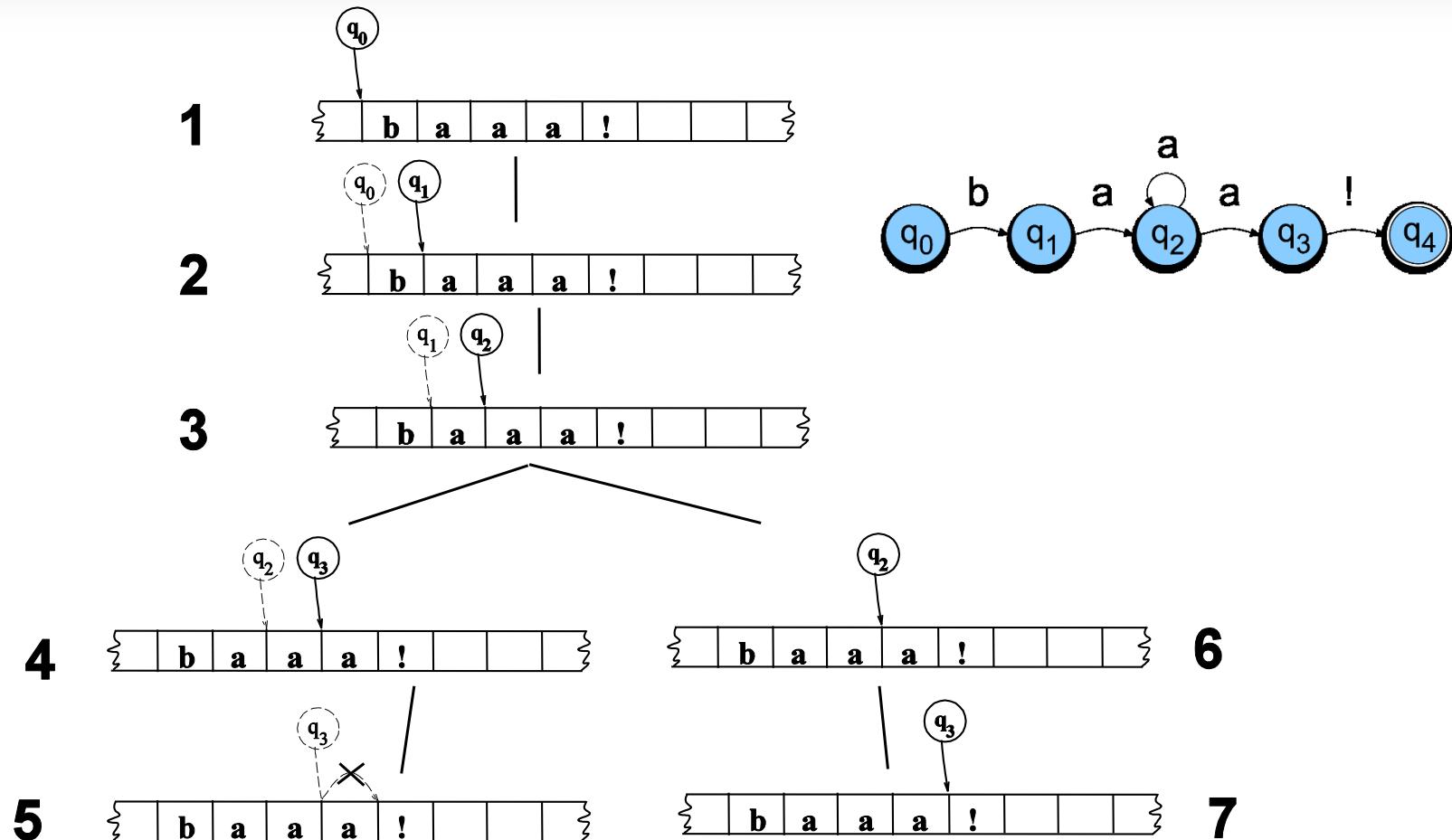
Example



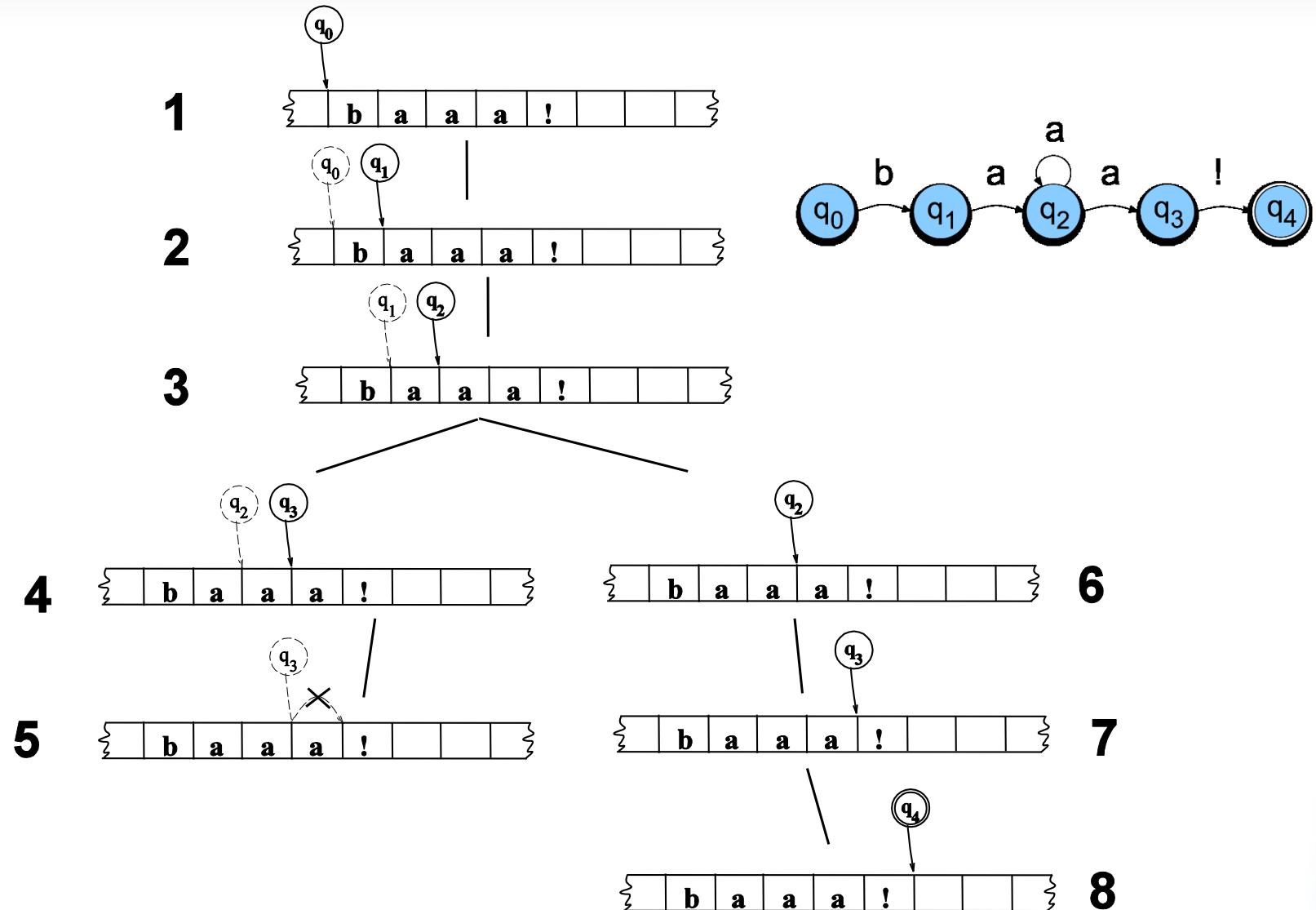
Example



Example



Example



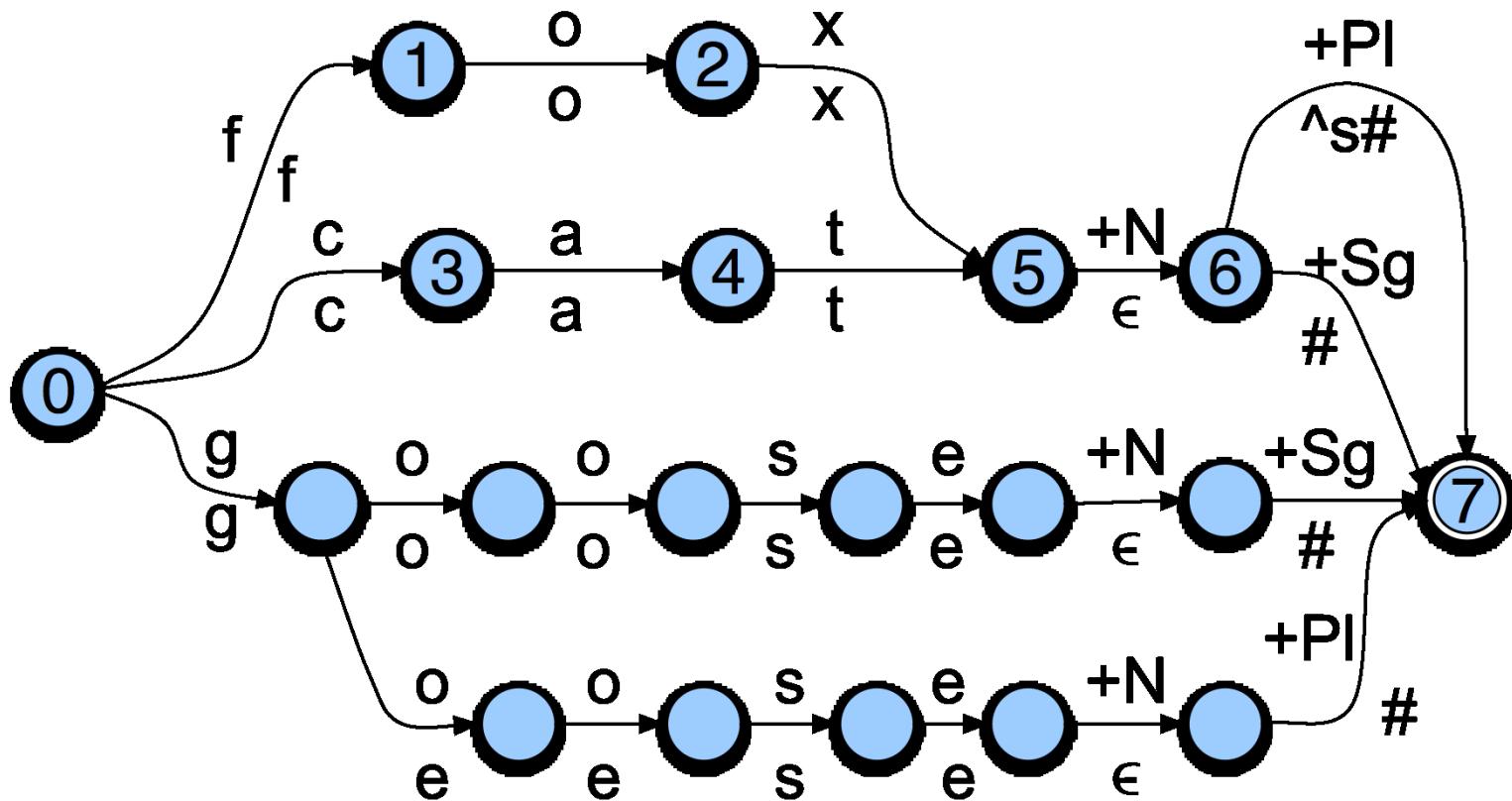
Key Points

- States in the search space are pairings of tape positions and states in the machine.
- By keeping track of as yet unexplored states, a recognizer can systematically explore all the paths through the machine given an input.

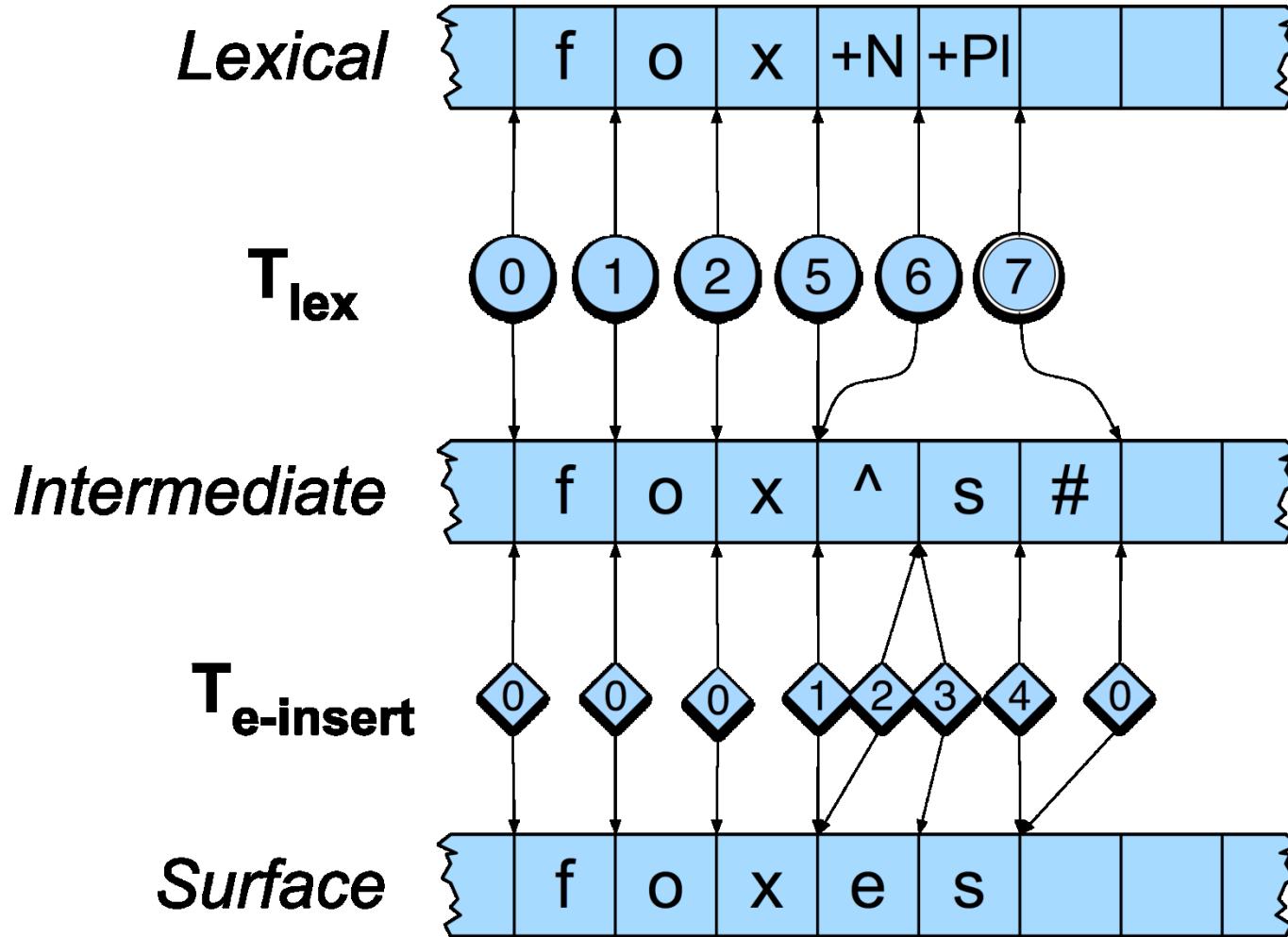
FSTs (Contd)

FST Fragment: Lexical to Intermediate

- ^ is morpheme boundary; # is word boundary



Putting Them Together



Practical Uses

- This kind of parsing is normally called morphological analysis
- Can be
 - An important stand-alone component of an application (spelling correction, information retrieval, part-of-speech tagging,...)
 - Or simply a link in a chain of processing (machine translation, parsing,...)

FST-based Tokenization

```
#!/usr/bin/perl

$letternumber = "[A-Za-z0-9]";
$notletter = "[^A-Za-z0-9]";
$alwayssep = "[\\?!()\";/\\[\\]]";
$clitic = "('':|-|'S'|'D'|'M'|'LL'|'RE|'VE|'N'T|'s|'d|'m|'ll|'re|'ve|'n't)";

$abbr{"Co."} = 1; $abbr{"Dr."} = 1; $abbr{"Jan."} = 1; $abbr{"Feb."} = 1;

while ($line = <>){ # read the next line from standard input

    # put whitespace around unambiguous separators
    $line =~ s/$alwayssep/ $& /g;

    # put whitespace around commas that aren't inside numbers
    $line =~ s/([^\0-9]),/$1 , /g;
    $line =~ s/,([^\0-9])/ , $1/g;

    # distinguish single quotes from apostrophes by
    # segmenting off single quotes not preceded by letter
    $line =~ s/'/$& /g;
    $line =~ s/($notletter)'/ $1 '/g;

    # segment off unambiguous word-final clitics and punctuation
    $line =~ s/$clitic$/ $& /g;
    $line =~ s/$clitic($notletter)/ $1 $2/g;

    # now deal with periods. For each possible word
    @possiblewords=split(/\s+/, $line);
    foreach $word (@possiblewords) {
        # if it ends in a period,
        if (($word =~ $letternumber\.) )
            && !($abbr{$word}) # and isn't on the abbreviation list
            # and isn't a sequence of letters and periods (U.S.)
            # and doesn't resemble an abbreviation (no vowels: Inc.)
            && !($word =~
                  ^([A-Za-z]\.([A-Za-z]\.)*|[A-Z][bcdfghj-nptvxz]+\.)$/) {
                # then segment off the period
                $word =~ s/\.$/ \./;
            }
        # expand clitics
        $word =~ s/'ve/have/;
        $word =~ s/'m/am/;
        print $word, " ";
    }
    print "\n";
}
```

Porter Stemmer (1980)

- Common algorithm for stemming English
- Conventions + 5 phases of reductions
 - ◆ phases applied sequentially
 - ◆ each phase consists of a set of commands
 - ◆ sample convention: *Of the rules in a compound command, select the one that applies to the longest suffix.*

Porter Stemmer (1980)

- Standard, very popular and usable **stemmer** (IR, IE) – identify a word's **stem**
- Sequence of cascaded rewrite rules, e.g.
 - ◆ IZE → ε (e.g. unionize → union)
 - ◆ CY → T (e.g. frequency → frequent)
 - ◆ ING → ε , if stem contains vowel (motoring → motor)
- Can be implemented as a lexicon-free FST (many implementations available on the web)
- <http://text-processing.com/demo/stem/>

Eliza

User₁: Men are all alike.

ELIZA₁: IN WHAT WAY

User₂: They're always bugging us about something or other.

ELIZA₂: CAN YOU THINK OF A SPECIFIC EXAMPLE

User₃: Well, my boyfriend made me come here.

ELIZA₃: YOUR BOYFRIEND MADE YOU COME HERE

User₄: He says I'm depressed much of the time.

ELIZA₄: I AM SORRY TO HEAR YOU ARE DEPRESSED

Eliza FST

```
s/.* I'M (depressed|sad) .*/I AM SORRY TO HEAR YOU ARE \1/  
s/.* I AM (depressed|sad) .*/WHY DO YOU THINK YOU ARE \1/  
s/.* all .*/IN WHAT WAY/  
s/.* always .*/CAN YOU THINK OF A SPECIFIC EXAMPLE/
```

RelNoun: Nominal Open IE

Constructions	Phrase	Extraction
Verb1	Francis Collins is the director of NIH	(Francis Collins; is the director of; NIH)
Verb2	the director of NIH is Francis Collins	(Francis Collins; is the director of; NIH)
Appositive1	Francis Collins, the director of NIH	(Francis Collins; [is] the director of; NIH)
Appositive2	the director of NIH, Francis Collins,	(Francis Collins; [is] the director of; NIH)
Appositive3	Francis Collins, the NIH director	(Francis Collins; [is] the director [of]; NIH)
AppositiveTitle	Francis Collins, the director,	(Francis Collins; [is]; the director)
CompoundNoun	<i>NIH director Francis Collins</i>	(Francis Collins; [is] director [of]; NIH)
Possessive	NIH's director Francis Collins	(Francis Collins; [is] director [of]; NIH)
PossessiveAppositive	NIH's director, Francis Collins	(Francis Collins; [is] director [of]; NIH)
AppositivePossessive	Francis Collins, NIH's director	(Francis Collins; [is] director [of]; NIH)
PossessiveVerb	NIH's director is Francis Collins	(Francis Collins; is director [of]; NIH)
VerbPossessive	Francis Collins is NIH's director	(Francis Collins; is director [of]; NIH)

Compound Noun Extraction

Baseline

- NIH Director Francis Collins

(Francis Collins, is the Director of, NIH)

- Challenges

- ◆ New York Banker Association

ORG NAMES

- ◆ German Chancellor Angela Merkel

DEMONYMS

- ◆ Prime Minister Modi

COMPOUND
RELATIONAL NOUNS

- ◆ GM Vice Chairman Bob Lutz

Rule-Based System

- Classifies and filters orgs
- List of demonyms
 - ◆ appropriate location conversion
- Bootstrap a list of relational noun *prefixes*
 - ◆ vice, ex, health, ...

Summing Up

- Regular expressions and FSAs can represent subsets of natural language as well as regular languages
 - ◆ Both representations may be difficult for humans to use for any real subset of a language
 - ◆ But quick, powerful and easy to use for small problems
- Finite state transducers and rules are common ways to incorporate linguistic ideas in NLP for small applications
- Particularly useful for no data setting

Finite-state methods for morphology

Julia Hockenmaier

Today's lecture

What are words? How many words are there?

What is the **structure of words**?

(in English, Chinese, Arabic,...)

Morphology: the area of linguistics that deals with this.

How can we identify the structure of words?

We need to build a **morphological analyzer** (parser).

We will use **finite-state transducers** for this task.

Finite-State Automata and Regular Languages

(Review)

Morphology: What is a word?

A Turkish word

uygarlaştıramadıklarımızdanmışsınızcasına
uygar_laş_tır_ama_dık_lar_ımız_dan_mış_sınız_casına

“as if you are among those whom we were not able to civilize (=cause to become civilized)”

uygar: *civilized*

laş: *become*

tır: *cause somebody to do something*

ama: *not able*

dık: *past participle*

lar: *plural*

ımız: *1st person plural possessive (our)*

dan: *among (ablative case)*

mış: *past*

sınız: *2nd person plural (you)*

casına: *as if (forms an adverb from a verb)*

K. Oflazer pc to J&M

Basic word classes (parts of speech)

Content words (open-class):

- Nouns: student, university, knowledge,...
- Verbs: write, learn, teach,...
- Adjectives: difficult, boring, hard,
- Adverbs: easily, repeatedly,...

Function words (closed-class):

- Prepositions: in, with, under,...
- Conjunctions: and, or,...
- Determiners: a, the, every,...

How many words are there?

The Unix command “`wc -w`” counts the words in a file.

```
> cat example.txt
```

This company isn't New York-based anymore

We moved to Chicago

```
> wc -w example.txt
```

10 example.txt

“`wc -w`” uses blanks to identify words:

This₁ company₂ isn't₃ New₄ York-based₅ anymore₆

We₇ moved₈ to₉ Chicago₁₀

Words aren't just defined by blanks

Problem 1: Compounding

“ice cream”, “website”, “web site”, “New York-based”

Problem 2: Other writing systems have no blanks

Chinese: 我开始写小说 = 我 开始 写 小说
 I start(ed) writing novel(s)

Problem 3: Clitics

English: “doesn’t”, “I’m”,

Italian: “dirglielo” = dir + gli(e) + lo
 tell + him + it

How many words are there?

Of course he wants to take the advanced course too.
He already took two beginners' courses.

This is a bad question. Did I mean:

How many **word tokens** are there?
(16 to 19, depending on how we count punctuation)

How many **word types** are there?
(i.e. How many different words are there?
Again, this depends on how you count, but it's
usually much less than the number of tokens)

How many words are there?

Of course he wants to take the advanced course too.
He already took two beginners' courses.

The same (underlying) word can take different forms:
course/courses, take/took

We distinguish concrete word forms (take, taking)
from abstract lemmas or dictionary forms (take)

Different words may be spelled/pronounced the same:
of course vs. advanced course
two vs. too

How many different words are there?

Inflection creates different forms of the same word:

Verbs: to be, being, I am, you are, he is, I was,

Nouns: one book, two books

Derivation creates different words from the same lemma:

grace ⇒ disgrace ⇒ disgraceful ⇒ disgracefully

Compounding combines two words into a new word:

cream ⇒ ice cream ⇒ ice cream cone ⇒ ice cream cone bakery

Word formation is productive:

New words are subject to all of these processes:

Google ⇒ Googler, to google, to ungoogle, to misgoogle, googlification, ungooglification, googlified, Google Maps, Google Maps service,...

Inflectional morphology in English

Verbs:

- Infinitive/present tense: walk, go
- 3rd person singular present tense (s-form): walks, goes
- Simple past: walked, went
- Past participle (ed-form): walked, gone
- Present participle (ing-form): walking, going

Nouns:

- Number: singular (book) vs. plural (books)
- Plural: books
- Possessive (~ genitive case): book's, books
- Personal pronouns inflect for person, number, gender, case:
I saw him; he saw me; you saw her; we saw them; they saw us.

Derivational morphology

Nominalization:

V + -ation: computerization

V+ -er: killer

Adj + -ness: fuzziness

Negation:

un-: undo, unseen, ...

mis-: mistake,...

Adjectivization:

V+ -able: doable

N + -al: national

Morphemes: stems, affixes

dis-grace-ful-ly
prefix-stem-suffix-suffix

Many word forms consist of a **stem** plus a number of **affixes** (*prefixes or suffixes*)

Infixes are inserted inside the stem.

Circumfixes (German gesehen) surround the stem

Morphemes: the smallest (meaningful/grammatical) parts of words.

Stems (grace) are often **free morphemes**.

Free morphemes can occur by themselves as words.

Affixes (dis-, -ful, -ly) are usually **bound morphemes**.

Bound morphemes have to combine with others to form words.

Morphemes and morphs

There are many *irregular* word forms:

- Plural nouns add -s to singular: book-books,
but: box-boxes, fly-flies, child-children
- Past tense verbs add -ed to infinitive: walk-walked,
but: like-liked, leap-leapt

Morphemes are abstract categories

Examples: plural morpheme, past tense morpheme

The same morpheme (e.g. for plural nouns) can be realized as different surface forms (morphs):

-s/-es/-ren

Allomorphs: two different realizations (-s/-es/-ren)
of the same underlying morpheme (plural)

Morphological parsing and generation

Morphological parsing

disgracefully
dis grace ful ly
prefix stem suffix suffix
NEG grace+N +ADJ +ADV

Morphological generation

Generate possible English words:

grace, graceful, gracefully
disgrace, disgraceful, disgracefully,
ungraceful, ungracefully,
undisgraceful, undisgracefully,...

Don't generate impossible English words:

*gracelyful, *gracefuly, *disungracefully,...

Review: Finite-State Automata and Regular Languages

Formal languages

An alphabet Σ is a set of symbols:

e.g. $\Sigma = \{a, b, c\}$

A string ω is a sequence of symbols, e.g $\omega = abcb$.

The empty string ε consists of zero symbols.

The Kleene closure Σ^* ('sigma star') is the (infinite) set of all strings that can be formed from Σ :

$\Sigma^* = \{\varepsilon, a, b, c, aa, ab, ba, aaa, \dots\}$

A language $L \subseteq \Sigma^*$ over Σ is also a set of strings.

Typically we only care about proper subsets of Σ^* ($L \subset \Sigma^*$).

Automata and languages

An **automaton** is an abstract model of a computer which reads an **input string**, and **changes its internal state** depending on the current input symbol. It can either **accept or reject** the input string.

Every automaton defines a language
(the set of strings it accepts).

Different automata define different language classes:

- **Finite-state** automata define **regular** languages
- **Pushdown** automata define **context-free** languages
- **Turing machines** define **recursively enumerable** languages

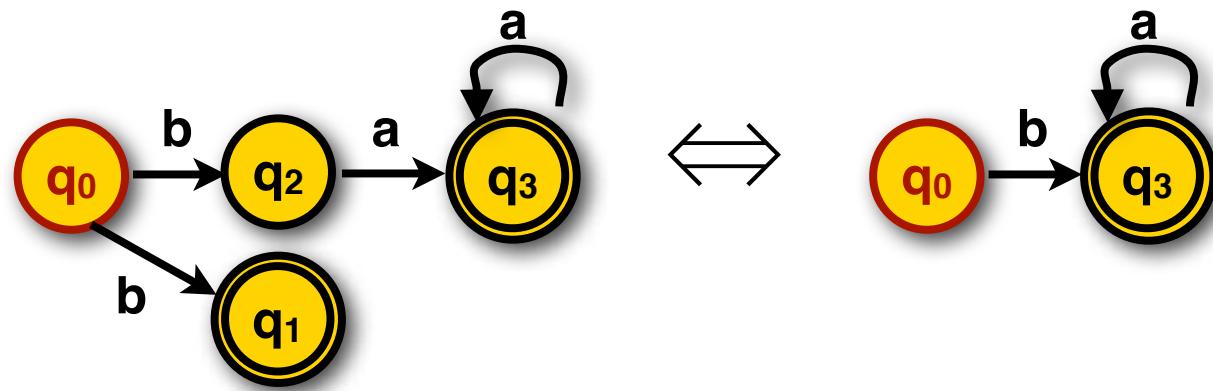
Finite State Automata (FSAs)

A finite-state automaton $M = \langle Q, \Sigma, q_0, F, \delta \rangle$ consists of:

- A finite set of states $Q = \{q_0, q_1, \dots, q_n\}$
- A finite alphabet Σ of input symbols (e.g. $\Sigma = \{a, b, c, \dots\}$)
- A designated start state $q_0 \in Q$
- A set of final states $F \subseteq Q$
- A transition function δ :
 - The transition function for a deterministic (D)FSA: $Q \times \Sigma \rightarrow Q$
$$\delta(q, w) = q' \quad \text{for } q, q' \in Q, w \in \Sigma$$
If the current state is q and the current input is w , go to q'
 - The transition function for a nondeterministic (N)FSA: $Q \times \Sigma \rightarrow 2^Q$
$$\delta(q, w) = Q' \quad \text{for } q \in Q, Q' \subseteq Q, w \in \Sigma$$
If the current state is q and the current input is w , go to any $q' \in Q'$

Finite State Automata (FSAs)

Every NFA can be transformed into an equivalent DFA:



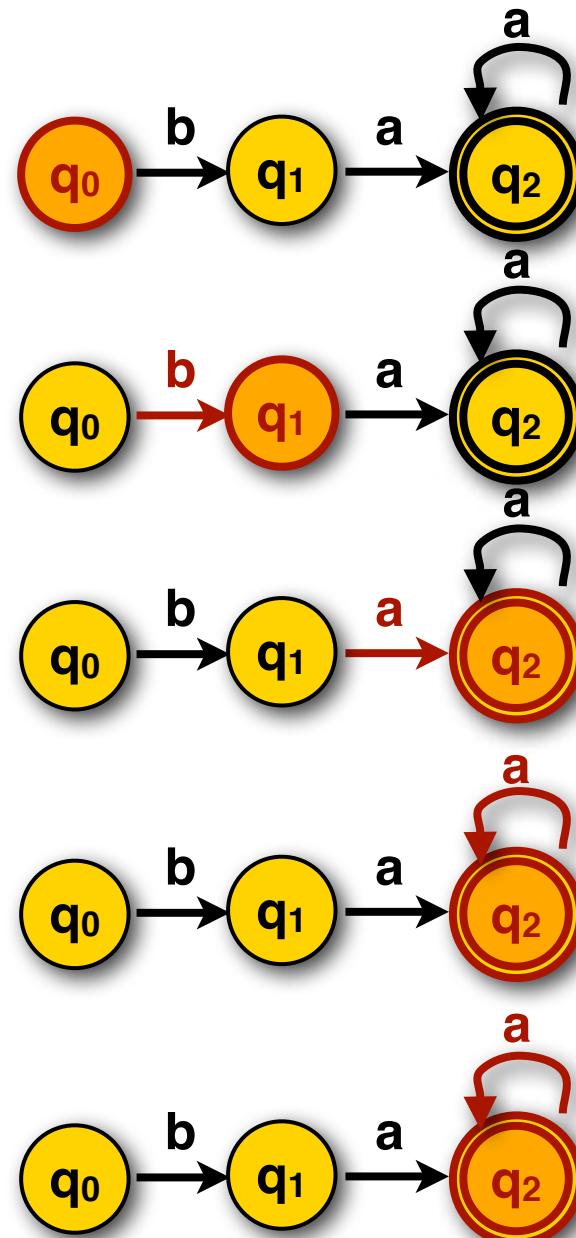
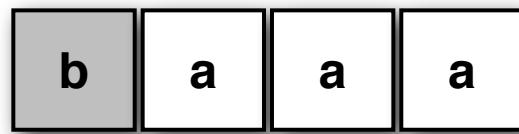
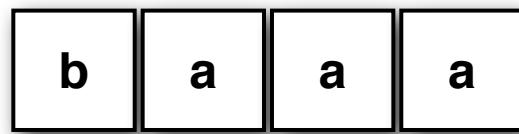
Recognition of a string w with a DFA is linear in the length of w

Finite-state automata define the class of **regular languages**

$L_1 = \{ a^n b^m \} = \{ ab, aab, abb, aaab, abb, \dots \}$ is a regular language,

$L_2 = \{ a^n b^n \} = \{ ab, aabb, aaabbb, \dots \}$ is not (it's context-free).

You cannot construct an FSA that accepts all the strings in L_2 and nothing else.



Regular Expressions

Simple patterns:

- **Standard characters** match themselves: ‘*a*’, ‘*I*’
- **Character classes**: ‘[*abc*]’, ‘[0-9]’, **negation**: ‘[^aeiou]’
(Predefined: \s (whitespace), \w (alphanumeric), etc.)
- **Any character** (except newline) is matched by ‘.’

Complex patterns: (e.g. ^[A-Z]([a-z])+\s)

- **Group**: ‘(...)'
- **Repetition**: 0 or more times: ‘*’, 1 or more times: ‘+’
- **Disjunction**: ‘...|...’
- **Beginning of line** ‘^’ and **end of line** ‘\$’

Python: Regular Expressions

```
>>> import re                                %% Import re package
>>> ex = re.compile('a.c')                    %% '...': reg.expression
>>> m = ex.search('ab')                      %% Does 'ab' contain ex?
>>> print m                                    %% No.

None

>>> m = ex.search('abc')                      %% Does 'abc' contain ex?
>>> print m                                    %% Yes.

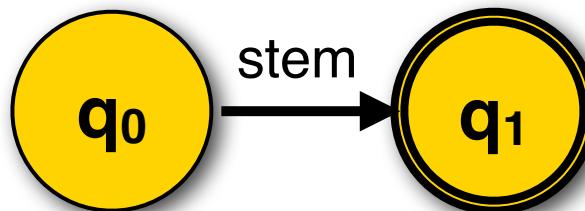
<_sre.SRE_Match object at 0x70640>
```

<http://docs.python.org/dev/howto/regex.html>
<http://docs.python.org/lib/module-re.html>

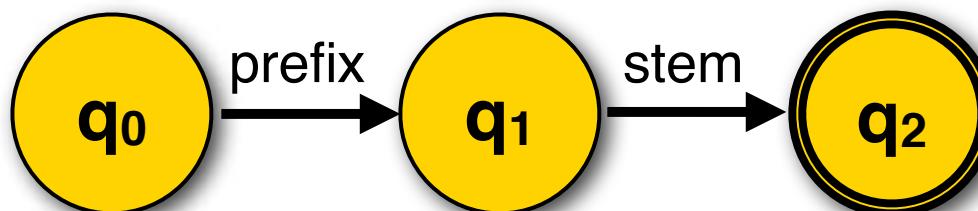
Finite-state methods for morphology

Finite state automata for morphology

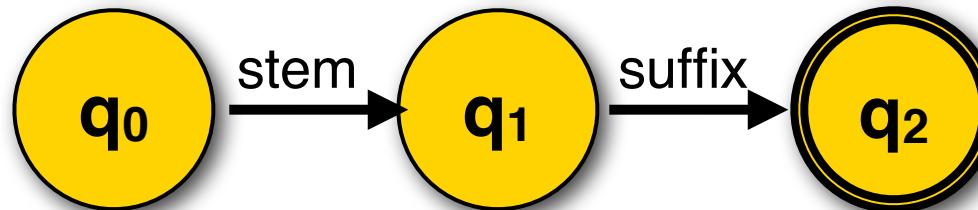
grace:



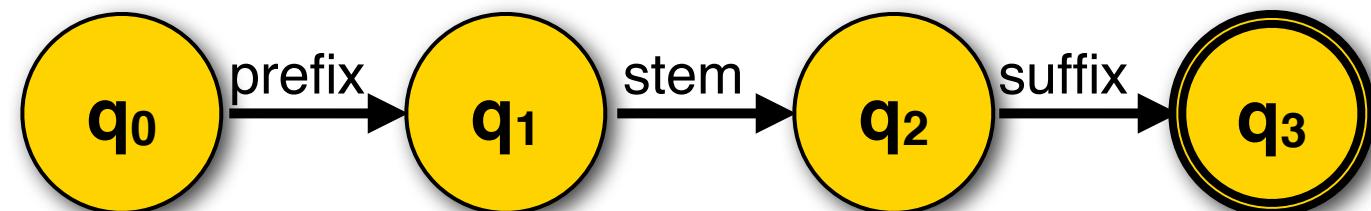
dis-grace:



grace-ful:

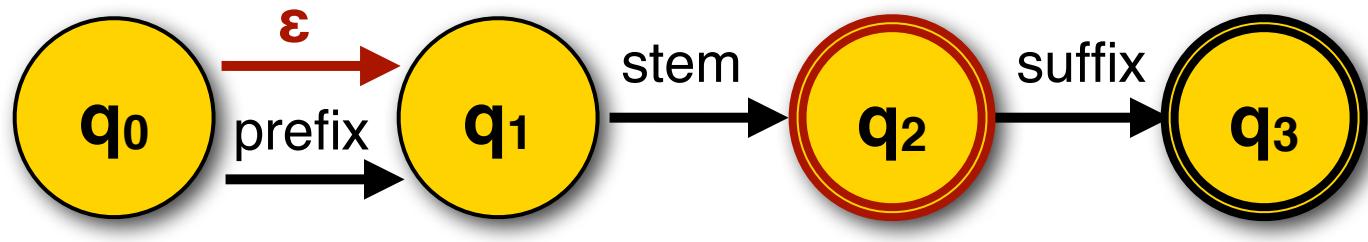


dis-grace-ful:



Union: merging automata

grace,
dis-grace,
grace-ful,
dis-grace-ful



Stem changes

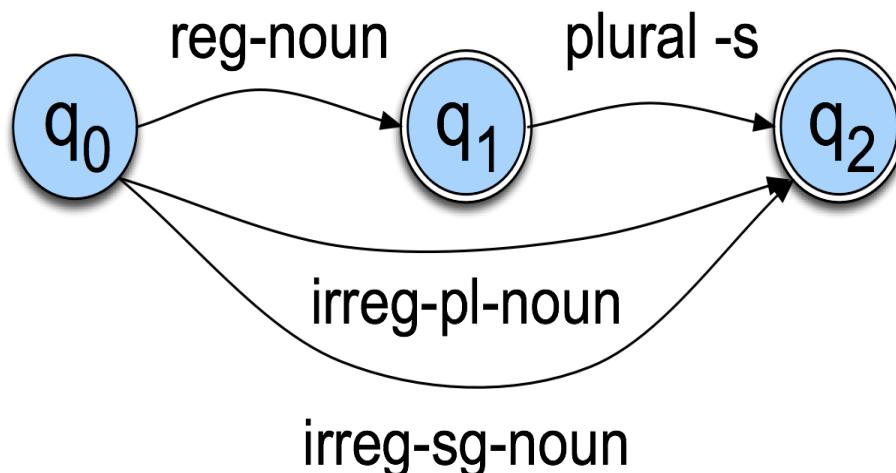
Some irregular words require stem changes:

Past tense verbs:

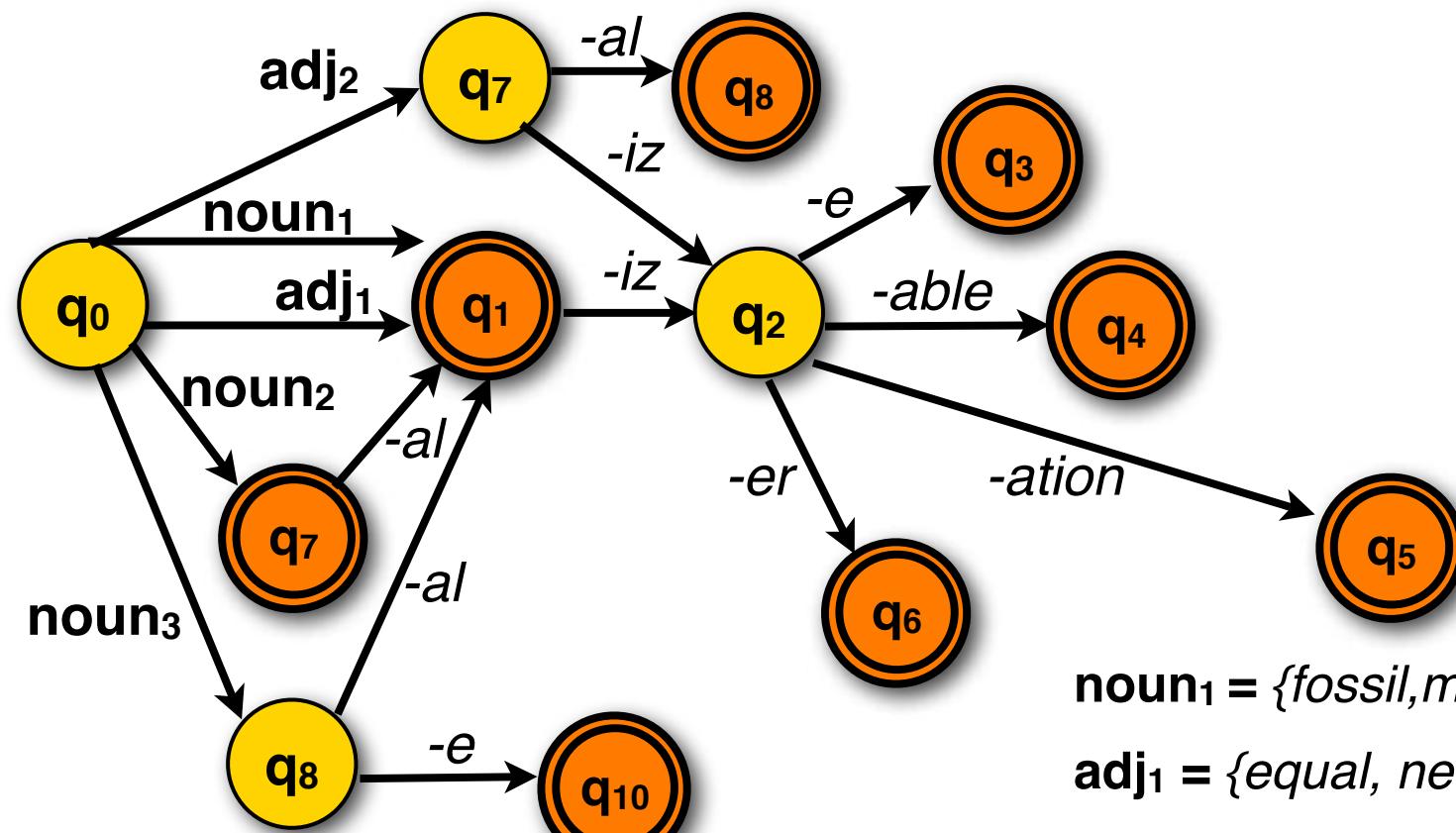
teach-taught, go-went, write-wrote

Plural nouns:

mouse-mice, foot-feet, wife-wives



FSAs for derivational morphology



noun₁ = {fossil, mineral, ...}

adj₁ = {equal, neutral}

adj₂ = {minim, maxim}

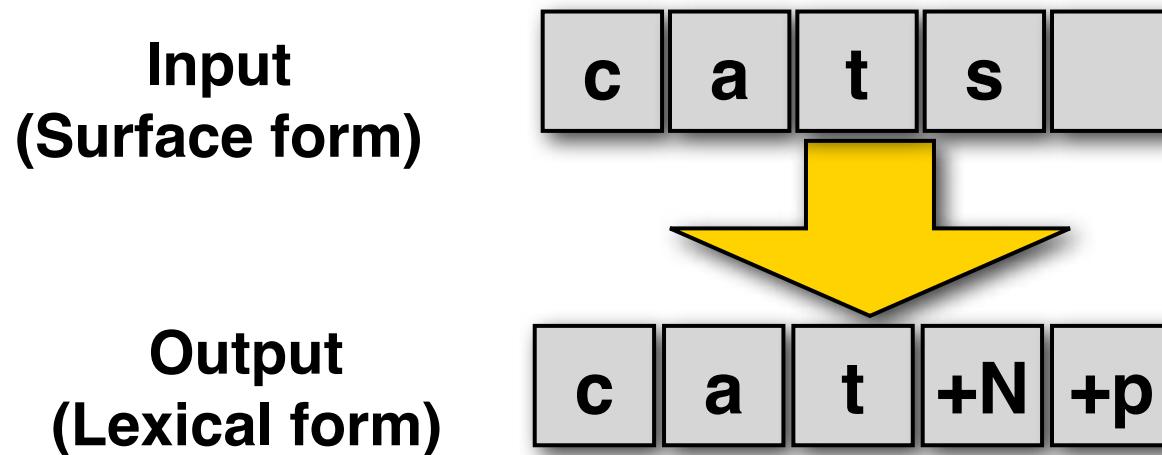
noun₂ = {nation, form, ...}

noun₃ = {natur, structur, ...}

Recognition vs. Analysis

FSAs can recognize (**accept**) a string, but they don't tell us its internal structure.

We need is a machine that maps (**transduces**) the input string into an output string that encodes its structure:



Finite-state transducers

A **finite-state transducer** $T = \langle Q, \Sigma, \Delta, q_0, F, \delta, \sigma \rangle$ consists of:

- A finite **set of states** $Q = \{q_0, q_1, \dots, q_n\}$
 - A finite alphabet Σ of **input symbols** (e.g. $\Sigma = \{a, b, c, \dots\}$)
 - A finite alphabet Δ of **output symbols** (e.g. $\Delta = \{+N, +pl, \dots\}$)
 - A designated **start state** $q_0 \in Q$
 - A set of **final states** $F \subseteq Q$
 - A **transition function** $\delta: Q \times \Sigma \rightarrow 2^Q$
 $\delta(q, w) = Q' \quad \text{for } q \in Q, Q' \subseteq Q, w \in \Sigma$
 - **An output function** $\sigma: Q \times \Sigma \rightarrow \Delta^*$
 $\sigma(q, w) = \omega \quad \text{for } q \in Q, w \in \Sigma, \omega \in \Delta^*$
- If the current state is q and the current input is w , write ω .

Finite-state transducers

An FST $T = L_{in} \times L_{out}$ defines a **relation between two regular languages L_{in} and L_{out} :**

$$L_{in} = \{\mathbf{cat}, \mathbf{cats}, \mathbf{fox}, \mathbf{foxes}, \dots\}$$
$$L_{out} = \{cat+N+sg, cat+N+pl, fox+N+sg, fox+N+PL \dots\}$$


$$T = \{ <\mathbf{cat}, cat+N+sg>, \\ <\mathbf{cats}, cat+N+pl>, \\ <\mathbf{fox}, fox+N+sg>, \\ <\mathbf{foxes}, fox+N+pl> \}$$

Some FST operations

Inversion T^{-1} :

The inversion (T^{-1}) of a transducer switches input and output labels.

*This can be used to switch from **parsing words** to **generating words**.*

Composition ($T \circ T'$): (*Cascade*)

Two transducers $T = L_1 \times L_2$ and $T' = L_2 \times L_3$ can be composed into a third transducer $T'' = L_1 \times L_3$.

*Sometimes **intermediate representations** are useful*

English spelling rules

English spelling (orthography) is funny:

The underlying morphemes (*plural-s*, etc.) can have different orthographic surface realizations (-s, -es)

Spelling changes at morpheme boundaries:

- E-insertion: fox +s = fox**e**s
- E-deletion: make**e** +ing = making

Intermediate representations

English plural -s: cat \Rightarrow cats dog \Rightarrow dogs
but: fox \Rightarrow foxes, bus \Rightarrow buses buzz \Rightarrow buzzes

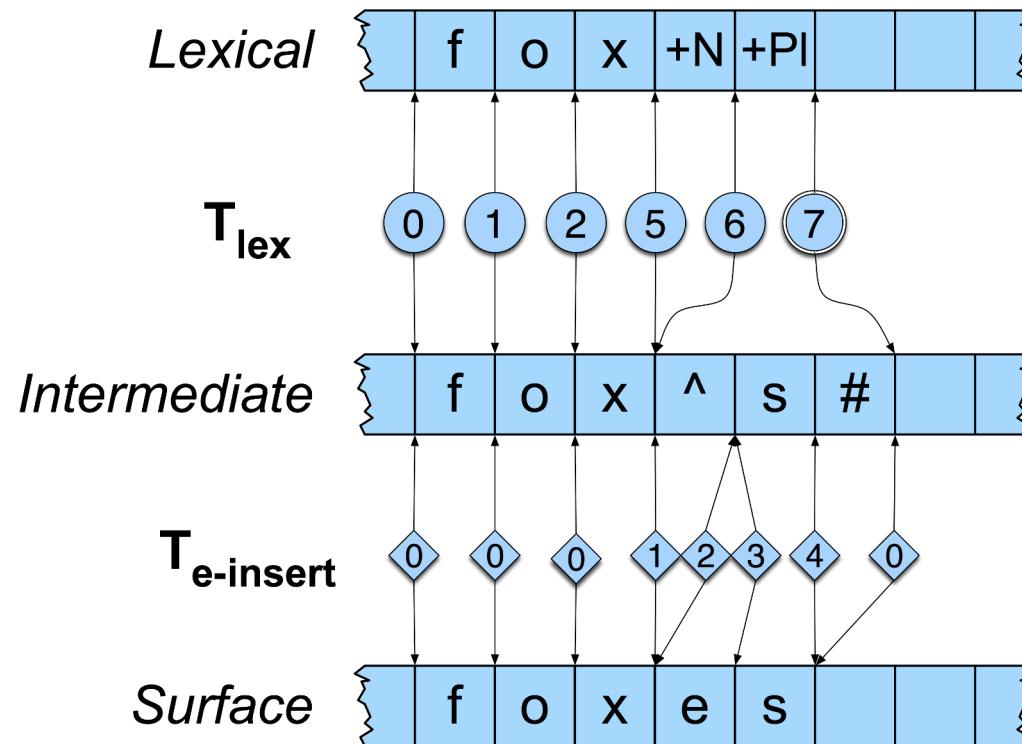
We define an **intermediate representation** which captures morpheme boundaries (^) and word boundaries (#):

<i>Lexicon:</i>	<i>cat+N+PL</i>	<i>fox+N+PL</i>
\Rightarrow <i>Intermediate representation:</i>	<i>cat^s#</i>	<i>fox^s#</i>
\Rightarrow <i>Surface string:</i>	<i>cats</i>	<i>foxes</i>

Intermediate-to-Surface Spelling Rule:

If plural '**s**' follows a morpheme ending in '**x**', '**z**' or '**s**', insert '**e**'.

FST composition/cascade:

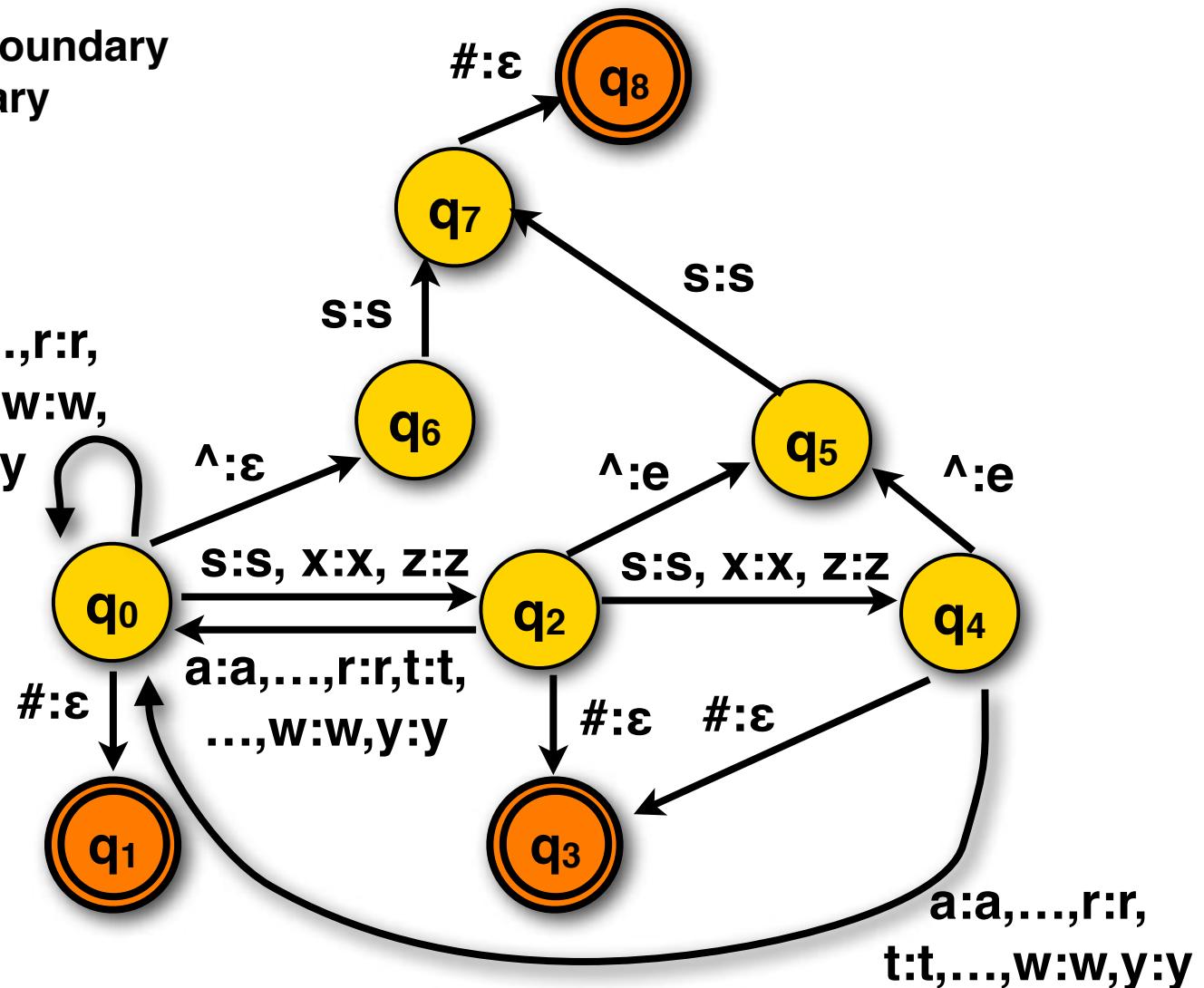


FST: intermediate to surface level

\wedge = morpheme boundary
 $\#$ = word boundary

Intermediate-to-Surface Spelling Rule:

If plural 's' follows a morpheme ending in 'x', 'z' or 's', insert 'e'.



Dealing with ambiguity

book: $book + N + sg$ or $book + V?$

Generating words is generally unambiguous, but
analyzing words often requires disambiguation.

Efficiency problem:

Not every nondeterministic FST can be translated into
a deterministic one!

A computer therapist?

Computer: TELL ME MORE ABOUT YOUR FAMILY

Human: My mother takes care of me.

Computer: WHO ELSE IN YOUR FAMILY TAKES CARE OF YOU

Human: My father.

Computer: YOUR FATHER

Human: You are like my father in some ways.

Computer: WHAT RESEMBLANCE DO YOU SEE

Human: You are not very aggressive but I think you don't want me to notice that.

Computer: WHAT MAKES YOU THINK I AM NOT VERY AGGRESSIVE

Human: You don't argue with me.

Computer: WHY DO YOU THINK I DON'T ARGUE WITH YOU

Human: You are afraid of me.

Weizenbaum (1966), ELIZA.

ELIZA as a FST cascade

Human: You don't argue with me.

Computer: WHY DO YOU THINK I DON'T ARGUE WITH YOU

1. Replace **you** with **I** and **me** with **you**:

I don't argue with you.

2. Replace <...> with **Why do you think <...>**:

Why do you think I don't argue with you.

What about compounds?

Compounds have hierarchical structure:

((ice cream) cone) bakery

not (ice ((cream cone) bakery))

((computer science) (graduate student))

not (computer ((science graduate) student))

We will need context-free grammars to capture this underlying structure.

Language Modeling

Introduction to N-grams

Probabilistic Language Models

Today's goal: assign a probability to a sentence

- Machine Translation:
 - $P(\text{high winds tonite}) > P(\text{large winds tonite})$
- Spell Correction

Why?

- The office is about fifteen minuets from my house
 - $P(\text{about fifteen minutes from}) > P(\text{about fifteen minuets from})$
- Speech Recognition
 - $P(\text{I saw a van}) \gg P(\text{eyes awe of an})$
- + Summarization, question-answering, etc., etc.!!

Probabilistic Language Modeling

Goal: compute the probability of a sentence or sequence of words:

$$P(W) = P(w_1, w_2, w_3, w_4, w_5 \dots w_n)$$

Related task: probability of an upcoming word:

$$P(w_5 | w_1, w_2, w_3, w_4)$$

A model that computes either of these:

$P(W)$ or $P(w_n | w_1, w_2 \dots w_{n-1})$ is called a **language model**.

Better: **the grammar** But **language model** or **LM** is standard

How to compute $P(W)$

How to compute this joint probability:

- $P(\text{its, water, is, so, transparent, that})$

Intuition: let's rely on the Chain Rule of Probability

Reminder: The Chain Rule

Recall the definition of conditional probabilities

$$p(B|A) = P(A,B)/P(A) \quad \text{Rewriting: } P(A,B) = P(A)P(B|A)$$

More variables:

$$P(A,B,C,D) = P(A)P(B|A)P(C|A,B)P(D|A,B,C)$$

The Chain Rule in General

$$P(x_1, x_2, x_3, \dots, x_n) = P(x_1)P(x_2|x_1)P(x_3|x_1, x_2)\dots P(x_n|x_1, \dots, x_{n-1})$$

The Chain Rule applied to compute joint probability of words in sentence

$$P(w_1 w_2 \dots w_n) = \prod_i P(w_i | w_1 w_2 \dots w_{i-1})$$

$P(\text{"its water is so transparent"}) =$

$P(\text{its}) \times P(\text{water} | \text{its}) \times P(\text{is} | \text{its water})$

$\times P(\text{so} | \text{its water is}) \times P(\text{transparent} | \text{its water is so})$

How to estimate these probabilities

Could we just count and divide?

$P(\text{the lits water is so transparent that}) =$

$$\frac{\text{Count}(\text{its water is so transparent that the})}{\text{Count}(\text{its water is so transparent that})}$$

No! Too many possible sentences!

We'll never see enough data for estimating these

Markov Assumption



Andrei Markov

Simplifying assumption:

$P(\text{the lit water is so transparent that}) \approx P(\text{the lit that})$

Or maybe

$P(\text{the lit water is so transparent that}) \approx P(\text{the transparent that})$

Markov Assumption

$$P(w_1 w_2 \dots w_n) \approx \prod_i P(w_i | w_{i-k} \dots w_{i-1})$$

In other words, we approximate each component in the product

$$P(w_i | w_1 w_2 \dots w_{i-1}) \approx P(w_i | w_{i-k} \dots w_{i-1})$$

Simplest case: Unigram model

$$P(w_1 w_2 \dots w_n) \approx \prod_i P(w_i)$$

Some automatically generated sentences from a unigram model

fifth, an, of, futures, the, an, incorporated, a,
a, the, inflation, most, dollars, quarter, in, is,
mass

thrift, did, eighty, said, hard, 'm, july, bullish

that, or, limited, the

Bigram model

- Condition on the previous word:

$$P(w_i | w_1 w_2 \dots w_{i-1}) \approx P(w_i | w_{i-1})$$

texaco, rose, one, in, this, issue, is, pursuing, growth, in,
a, boiler, house, said, mr., gurria, mexico, 's, motion,
control, proposal, without, permission, from, five, hundred,
fifty, five, yen

outside, new, car, parking, lot, of, the, agreement, reached
this, would, be, a, record, november

N-gram models

We can extend to trigrams, 4-grams, 5-grams

In general this is an insufficient model of language

- because language has **long-distance dependencies**:

“The computer which I had just put into the machine room
on the fifth floor crashed.”

But we can often get away with N-gram models

Language Modeling

Introduction to N-grams

Language Modeling

Estimating N-gram Probabilities

Estimating bigram probabilities

The Maximum Likelihood Estimate

$$P(w_i | w_{i-1}) = \frac{\text{count}(w_{i-1}, w_i)}{\text{count}(w_{i-1})}$$

$$P(w_i | w_{i-1}) = \frac{c(w_{i-1}, w_i)}{c(w_{i-1})}$$

An example

$$P(w_i | w_{i-1}) = \frac{c(w_{i-1}, w_i)}{c(w_{i-1})}$$

< s > I am Sam < /s >

< s > Sam I am < /s >

< s > I do not like green eggs and ham < /s >

$$P(\text{I} | \text{<s>}) = \frac{2}{3} = .67$$

$$P(\text{Sam} | \text{<s>}) = \frac{1}{3} = .33$$

$$P(\text{am} | \text{I}) = \frac{2}{3} = .67$$

$$P(\text{</s>} | \text{Sam}) = \frac{1}{2} = 0.5$$

$$P(\text{Sam} | \text{am}) = \frac{1}{2} = .5$$

$$P(\text{do} | \text{I}) = \frac{1}{3} = .33$$

More examples: Berkeley Restaurant Project sentences

can you tell me about any good cantonese restaurants close by
mid priced thai food is what i'm looking for
tell me about chez panisse
can you give me a listing of the kinds of food that are available
i'm looking for a good place to eat breakfast
when is caffe venezia open during the day

Raw bigram counts

Out of 9222 sentences

	i	want	to	eat	chinese	food	lunch	spend
i	5	827	0	9	0	0	0	2
want	2	0	608	1	6	6	5	1
to	2	0	4	686	2	0	6	211
eat	0	0	2	0	16	2	42	0
chinese	1	0	0	0	0	82	1	0
food	15	0	15	0	1	4	0	0
lunch	2	0	0	0	0	1	0	0
spend	1	0	1	0	0	0	0	0

Raw bigram probabilities

Normalize by unigrams:

i	want	to	eat	chinese	food	lunch	spend
2533	927	2417	746	158	1093	341	278

Result:

	i	want	to	eat	chinese	food	lunch	spend
i	0.002	0.33	0	0.0036	0	0	0	0.00079
want	0.0022	0	0.66	0.0011	0.0065	0.0065	0.0054	0.0011
to	0.00083	0	0.0017	0.28	0.00083	0	0.0025	0.087
eat	0	0	0.0027	0	0.021	0.0027	0.056	0
chinese	0.0063	0	0	0	0	0.52	0.0063	0
food	0.014	0	0.014	0	0.00092	0.0037	0	0
lunch	0.0059	0	0	0	0	0.0029	0	0
spend	0.0036	0	0.0036	0	0	0	0	0

Bigram estimates of sentence probabilities

$P(< s > \mid \text{I want english food } < /s >) =$

$P(\text{I} \mid < s >)$

$\times P(\text{want} \mid \text{I})$

$\times P(\text{english} \mid \text{want})$

$\times P(\text{food} \mid \text{english})$

$\times P(< /s > \mid \text{food})$

$= .000031$

What kinds of knowledge?

$P(\text{english} \mid \text{want}) = .0011$

$P(\text{chinese} \mid \text{want}) = .0065$

$P(\text{to} \mid \text{want}) = .66$

$P(\text{eat} \mid \text{to}) = .28$

$P(\text{food} \mid \text{to}) = 0$

$P(\text{want} \mid \text{spend}) = 0$

$P(\text{i} \mid \langle s \rangle) = .25$

Practical Issues

We do everything in log space

- Avoid underflow
- (also adding is faster than multiplying)

$$\log(p_1 \times p_2 \times p_3 \times p_4) = \log p_1 + \log p_2 + \log p_3 + \log p_4$$

Language Modeling Toolkits

SRILM

- <http://www.speech.sri.com/projects/srilm/>

KenLM

- <https://kheafield.com/code/kenlm/>

Google N-Gram Release, August 2006

AUG

3

All Our N-gram are Belong to You

Posted by Alex Franz and Thorsten Brants, Google Machine Translation Team

Here at Google Research we have been using word [n-gram models](#) for a variety of R&D projects,

...

That's why we decided to share this enormous dataset with everyone. We processed 1,024,908,267,229 words of running text and are publishing the counts for all 1,176,470,663 five-word sequences that appear at least 40 times. There are 13,588,391 unique words, after discarding words that appear less than 200 times.

Google N-Gram Release

serve as the incoming 92
serve as the incubator 99
serve as the independent 794
serve as the index 223
serve as the indication 72
serve as the indicator 120
serve as the indicators 45
serve as the indispensable 111
serve as the indispensible 40
serve as the individual 234

Google Book N-grams

<http://ngrams.googlecode.com/>

Language Modeling

Estimating N-gram Probabilities

Language Modeling

Evaluation and Perplexity

Evaluation: How good is our model?

Does our language model prefer good sentences to bad ones?

- Assign higher probability to “real” or “frequently observed” sentences
 - Than “ungrammatical” or “rarely observed” sentences?

We train parameters of our model on a **training set**.

We test the model’s performance on data we haven’t seen.

- A **test set** is an unseen dataset that is different from our training set, totally unused.
- An **evaluation metric** tells us how well our model does on the test set.

Extrinsic evaluation of N-gram models

Best evaluation for comparing models A and B

- Put each model in a task
 - spelling corrector, speech recognizer, MT system
- Run the task, get an accuracy for A and for B
 - How many misspelled words corrected properly
 - How many words translated correctly
- Compare accuracy for A and B

Difficulty of extrinsic (in-vivo) evaluation of N-gram models

Extrinsic evaluation

- Time-consuming; can take days or weeks

So

- Sometimes use **intrinsic** evaluation: **perplexity**
- Bad approximation
 - unless the test data looks just like the training data
 - **So generally only useful in pilot experiments**
 - But is helpful to think about.

Intuition of Perplexity

The Shannon Game:

- How well can we predict the next word?

I always order pizza with cheese and _____

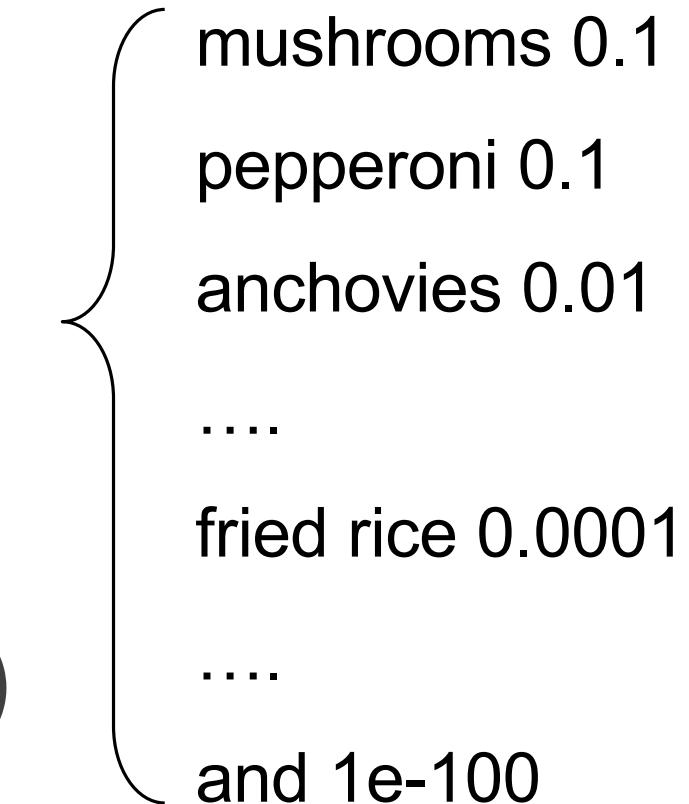
The 33rd President of the US was _____

I saw a _____

- Unigrams are terrible at this game. (Why?)

A better model of a text

- is one which assigns a higher probability to the word that actually occurs



Perplexity

The best language model is one that best predicts an unseen test set

- Gives the highest $P(\text{sentence})$

Perplexity is the inverse probability of the test set, normalized by the number of words:

$$PP(W) = P(w_1 w_2 \dots w_N)^{-\frac{1}{N}}$$

$$= \sqrt[N]{\frac{1}{P(w_1 w_2 \dots w_N)}}$$

Chain rule:

$$PP(W) = \sqrt[N]{\prod_{i=1}^N \frac{1}{P(w_i | w_1 \dots w_{i-1})}}$$

For bigrams:

$$PP(W) = \sqrt[N]{\prod_{i=1}^N \frac{1}{P(w_i | w_{i-1})}}$$

Minimizing perplexity is the same as maximizing probability

The Shannon Game intuition for perplexity

From Josh Goodman

Perplexity is weighted equivalent branching factor

How hard is the task of recognizing digits '0,1,2,3,4,5,6,7,8,9'

- Perplexity 10

How hard is recognizing (30,000) names at Microsoft.

- Perplexity = 30,000

Let's imagine a call-routing phone system gets 120K calls and has to recognize

- "Operator" (let's say this occurs 1 in 4 calls)
- "Sales" (1 in 4)
- "Technical Support" (1 in 4)
- 30,000 different names (each name occurring 1 time in the 120K calls)
- What is the perplexity? Next slide

The Shannon Game intuition for perplexity

Josh Goodman: imagine a call-routing phone system gets 120K calls and has to recognize

- "Operator" (let's say this occurs 1 in 4 calls)
- "Sales" (1 in 4)
- "Technical Support" (1 in 4)
- 30,000 different names (each name occurring 1 time in the 120K calls)

We get the perplexity of this sequence of length 120K by first multiplying 120K probabilities (90K of which are 1/4 and 30K of which are 1/120K), and then taking the inverse 120,000th root:

$$\text{Perp} = (\frac{1}{4} * \frac{1}{4} * \frac{1}{4} * \frac{1}{4} * \frac{1}{4} * \dots * \frac{1}{120K} * \frac{1}{120K} * \dots)^{(-1/120K)}$$

But this can be arithmetically simplified to just $N = 4$: the operator (1/4), the sales (1/4), the tech support (1/4), and the 30,000 names (1/120,000):

$$\text{Perplexity} = ((\frac{1}{4} * \frac{1}{4} * \frac{1}{4} * \frac{1}{4})^{(-1/4)} = 52.6$$

Perplexity as branching factor

Let's suppose a sentence consisting of random digits

What is the perplexity of this sentence according to a model that assign $P=1/10$ to each digit?

$$\begin{aligned} \text{PP}(W) &= P(w_1 w_2 \dots w_N)^{-\frac{1}{N}} \\ &= \left(\frac{1}{10}\right)^{-\frac{1}{N}} \\ &= \frac{1}{10}^{-1} \\ &= 10 \end{aligned}$$

Lower perplexity = better model

Training 38 million words, test 1.5 million words, WSJ

N-gram Order	Unigram	Bigram	Trigram
Perplexity	962	170	109

Language Modeling

Evaluation and Perplexity

Language Modeling

Generalization and zeros

The Shannon Visualization Method

Choose a random bigram

($\langle s \rangle$, w) according to its probability

Now choose a random bigram
(w, x) according to its probability

And so on until we choose $\langle /s \rangle$

Then string the words together

$\langle s \rangle$ I
I want
want to
to eat
eat Chinese
Chinese food
food $\langle /s \rangle$
I want to eat Chinese food

Approximating Shakespeare

1
gram

–To him swallowed confess hear both. Which. Of save on trail for are ay device and
rote life have
–Hill he late speaks; or! a more to leg less first you enter

2
gram

–Why dost stand forth thy canopy, forsooth; he is this palpable hit the King Henry. Live
king. Follow.
–What means, sir. I confess she? then all sorts, he is trim, captain.

3
gram

–Fly, and will rid me these news of price. Therefore the sadness of parting, as they say,
'tis done.
–This shall forbid it should be branded, if renown made it empty.

4
gram

–King Henry. What! I will go seek the traitor Gloucester. Exeunt some of the watch. A
great banquet serv'd in;
–It cannot be but so.

Shakespeare as corpus

$N=884,647$ tokens, $V=29,066$

Shakespeare produced 300,000 bigram types out of $V^2 = 844$ million possible bigrams.

- So 99.96% of the possible bigrams were never seen (have zero entries in the table)

Quadrigrams worse: What's coming out looks like Shakespeare because it *is* Shakespeare

The Wall Street Journal is not Shakespeare (no offense)

- 1 gram Months the my and issue of year foreign new exchange's september were recession exchange new endorsed a acquire to six executives
- 2 gram Last December through the way to preserve the Hudson corporation N. B. E. C. Taylor would seem to complete the major central planners one point five percent of U. S. E. has already old M. X. corporation of living on information such as more frequently fishing to keep her
- 3 gram They also point to ninety nine point six billion dollars from two hundred four oh six three percent of the rates of interest stores as Mexico and Brazil on market conditions

Can you guess the training set author of the LM that generated these random 3-gram sentences?

They also point to ninety nine point six billion dollars from two hundred four oh six three percent of the rates of interest stores as Mexico and gram Brazil on market conditions

This shall forbid it should be branded, if renown made it empty.

“You are uniformly charming!” cried he, with a smile of associating and now and then I bowed and they perceived a chaise and four to wish for.

The perils of overfitting

N-grams only work well for word prediction if the test corpus looks like the training corpus

- In real life, it often doesn’t
- We need to train robust models that generalize!
- One kind of generalization: Zeros!
- Things that don’t ever occur in the training set
 - But occur in the test set

Zeros

Training set:

- ... denied the allegations
- ... denied the reports
- ... denied the claims
- ... denied the request

- Test set
 - ... denied the offer
 - ... denied the loan

$$P(\text{"offer"} \mid \text{denied the}) = 0$$

Zero probability bigrams

Bigrams with zero probability

- mean that we will assign 0 probability to the test set!

And hence we cannot compute perplexity (can't divide by 0)!

Language Modeling

Generalization and zeros

Language Modeling

Smoothing: Add-one
(Laplace) smoothing

The intuition of smoothing (from Dan Klein)

When we have sparse statistics:

$P(w | \text{denied the})$

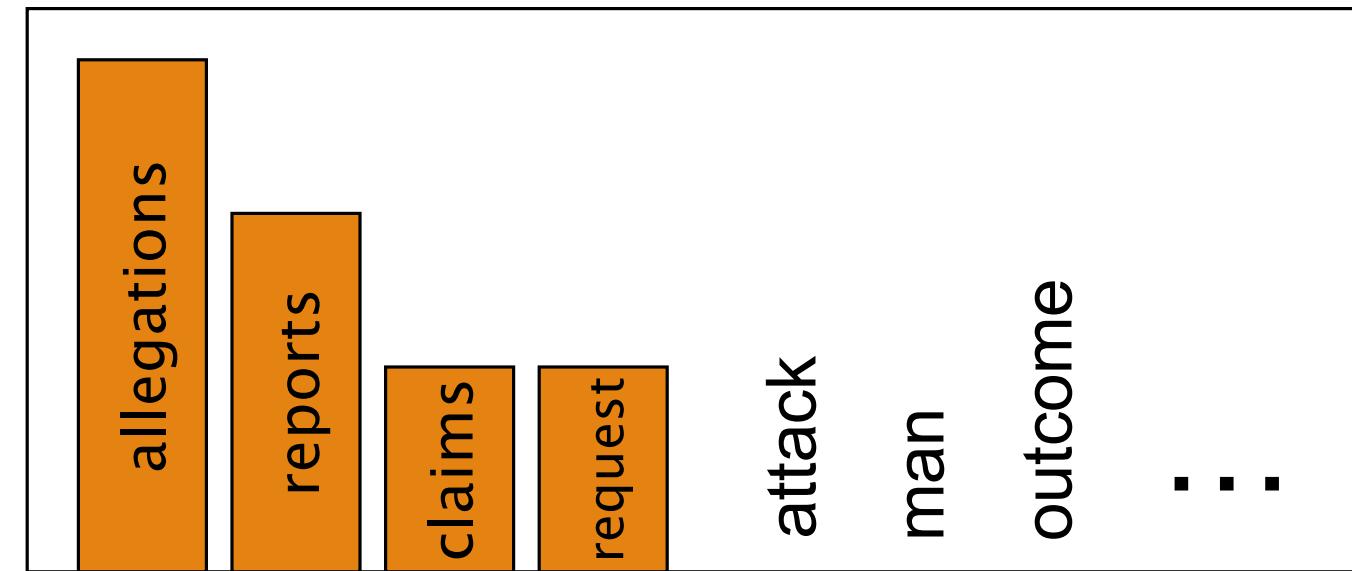
3 allegations

2 reports

1 claims

1 request

7 total



Steal probability mass to generalize better

$P(w | \text{denied the})$

2.5 allegations

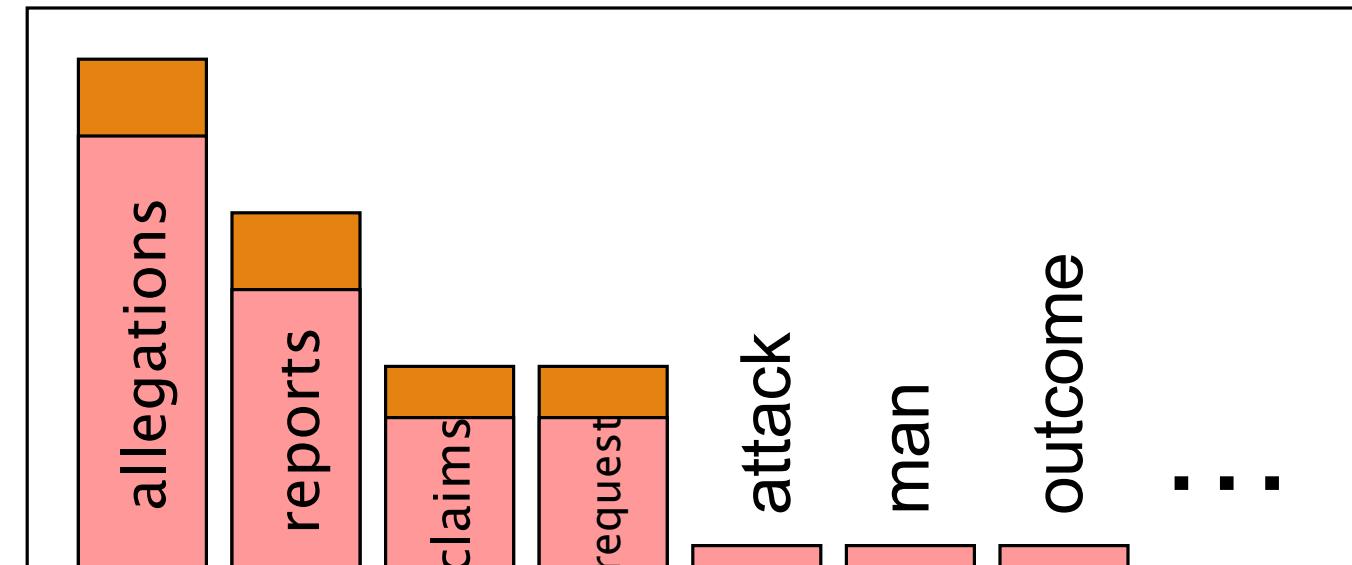
1.5 reports

0.5 claims

0.5 request

2 other

7 total



Add-one estimation

Also called Laplace smoothing

Pretend we saw each word one more time than we did
Just add one to all the counts!

MLE estimate:

$$P_{MLE}(w_i | w_{i-1}) = \frac{c(w_{i-1}, w_i)}{c(w_{i-1})}$$

Add-1 estimate:

$$P_{Add-1}(w_i | w_{i-1}) = \frac{c(w_{i-1}, w_i) + 1}{c(w_{i-1}) + V}$$

Maximum Likelihood Estimates

The maximum likelihood estimate

- of some parameter of a model M from a training set T
- maximizes the likelihood of the training set T given the model M

Suppose the word “bagel” occurs 400 times in a corpus of a million words

What is the probability that a random word from some other text will be “bagel”?

MLE estimate is $400/1,000,000 = .0004$

This may be a bad estimate for some other corpus

- But it is the **estimate** that makes it **most likely** that “bagel” will occur 400 times in a million word corpus.

Berkeley Restaurant Corpus: Laplace smoothed bigram counts

	i	want	to	eat	chinese	food	lunch	spend
i	6	828	1	10	1	1	1	3
want	3	1	609	2	7	7	6	2
to	3	1	5	687	3	1	7	212
eat	1	1	3	1	17	3	43	1
chinese	2	1	1	1	1	83	2	1
food	16	1	16	1	2	5	1	1
lunch	3	1	1	1	1	2	1	1
spend	2	1	2	1	1	1	1	1

Laplace-smoothed bigrams

$$P^*(w_n | w_{n-1}) = \frac{C(w_{n-1} w_n) + 1}{C(w_{n-1}) + V}$$

	i	want	to	eat	chinese	food	lunch	spend
i	0.0015	0.21	0.00025	0.0025	0.00025	0.00025	0.00025	0.00075
want	0.0013	0.00042	0.26	0.00084	0.0029	0.0029	0.0025	0.00084
to	0.00078	0.00026	0.0013	0.18	0.00078	0.00026	0.0018	0.055
eat	0.00046	0.00046	0.0014	0.00046	0.0078	0.0014	0.02	0.00046
chinese	0.0012	0.00062	0.00062	0.00062	0.00062	0.052	0.0012	0.00062
food	0.0063	0.00039	0.0063	0.00039	0.00079	0.002	0.00039	0.00039
lunch	0.0017	0.00056	0.00056	0.00056	0.00056	0.0011	0.00056	0.00056
spend	0.0012	0.00058	0.0012	0.00058	0.00058	0.00058	0.00058	0.00058

Reconstituted counts

$$c^*(w_{n-1}w_n) = \frac{[C(w_{n-1}w_n) + 1] \times C(w_{n-1})}{C(w_{n-1}) + V}$$

	i	want	to	eat	chinese	food	lunch	spend
i	3.8	527	0.64	6.4	0.64	0.64	0.64	1.9
want	1.2	0.39	238	0.78	2.7	2.7	2.3	0.78
to	1.9	0.63	3.1	430	1.9	0.63	4.4	133
eat	0.34	0.34	1	0.34	5.8	1	15	0.34
chinese	0.2	0.098	0.098	0.098	0.098	8.2	0.2	0.098
food	6.9	0.43	6.9	0.43	0.86	2.2	0.43	0.43
lunch	0.57	0.19	0.19	0.19	0.19	0.38	0.19	0.19
spend	0.32	0.16	0.32	0.16	0.16	0.16	0.16	0.16

Compare with raw bigram counts

	i	want	to	eat	chinese	food	lunch	spend
i	5	827	0	9	0	0	0	2
want	2	0	608	1	6	6	5	1
to	2	0	4	686	2	0	6	211
eat	0	0	2	0	16	2	42	0
chinese	1	0	0	0	0	82	1	0
food	15	0	15	0	1	4	0	0
lunch	2	0	0	0	0	1	0	0
spend	1	0	1	0	0	0	0	0

	i	want	to	eat	chinese	food	lunch	spend
i	3.8	527	0.64	6.4	0.64	0.64	0.64	1.9
want	1.2	0.39	238	0.78	2.7	2.7	2.3	0.78
to	1.9	0.63	3.1	430	1.9	0.63	4.4	133
eat	0.34	0.34	1	0.34	5.8	1	15	0.34
chinese	0.2	0.098	0.098	0.098	0.098	8.2	0.2	0.098
food	6.9	0.43	6.9	0.43	0.86	2.2	0.43	0.43
lunch	0.57	0.19	0.19	0.19	0.19	0.38	0.19	0.19
spend	0.32	0.16	0.32	0.16	0.16	0.16	0.16	0.16

Add-1 estimation is a blunt instrument

So add-1 isn't used for N-grams:

- We'll see better methods

But add-1 is used to smooth other NLP models

- For text classification
- In domains where the number of zeros isn't so huge.

Language Modeling

Smoothing: Add-one (Laplace) smoothing

Language Modeling

Interpolation, Backoff, and Web-Scale LMs

Backoff and Interpolation

Sometimes it helps to use **less context**

- Condition on less context for contexts you haven't learned much about

Backoff:

- use trigram if you have good evidence,
- otherwise bigram, otherwise unigram

Interpolation:

- mix unigram, bigram, trigram

Interpolation works better

Linear Interpolation

Simple interpolation

$$\begin{aligned}\hat{P}(w_n | w_{n-2} w_{n-1}) = & \lambda_1 P(w_n | w_{n-2} w_{n-1}) \\ & + \lambda_2 P(w_n | w_{n-1}) \\ & + \lambda_3 P(w_n)\end{aligned}$$

$$\sum_i \lambda_i = 1$$

Lambdas conditional on context:

$$\begin{aligned}\hat{P}(w_n | w_{n-2} w_{n-1}) = & \lambda_1(w_{n-2}^{n-1}) P(w_n | w_{n-2} w_{n-1}) \\ & + \lambda_2(w_{n-2}^{n-1}) P(w_n | w_{n-1}) \\ & + \lambda_3(w_{n-2}^{n-1}) P(w_n)\end{aligned}$$

How to set the lambdas?

Use a **held-out** corpus

Training Data

Held-Out
Data

Test
Data

Choose λ s to maximize the probability of held-out data:

- Fix the N-gram probabilities (on the training data)
- Then search for λ s that give largest probability to held-out set:

$$\log P(w_1 \dots w_n | M(\lambda_1 \dots \lambda_k)) = \sum_i \log P_{M(\lambda_1 \dots \lambda_k)}(w_i | w_{i-1})$$

Unknown words: Open versus closed vocabulary tasks

If we know all the words in advanced

- Vocabulary V is fixed
- Closed vocabulary task

Often we don't know this

- **Out Of Vocabulary** = OOV words
- Open vocabulary task

Instead: create an unknown word token `<UNK>`

- Training of `<UNK>` probabilities
 - Create a fixed lexicon L of size V
 - At text normalization phase, any training word not in L changed to `<UNK>`
 - Now we train its probabilities like a normal word
- At decoding time
 - If text input: Use UNK probabilities for any word not in training

Huge web-scale n-grams

How to deal with, e.g., Google N-gram corpus

Pruning

- Only store N-grams with count > threshold.
 - Remove singletons of higher-order n-grams
- Entropy-based pruning

Efficiency

- Efficient data structures like tries
- Bloom filters: approximate language models
- Store words as indexes, not strings
 - Use Huffman coding to fit large numbers of words into two bytes
- Quantize probabilities (4-8 bits instead of 8-byte float)

Smoothing for Web-scale N-grams

“Stupid backoff” (Brants *et al.* 2007)

No discounting, just use relative frequencies

$$S(w_i | w_{i-k+1}^{i-1}) = \begin{cases} \frac{\text{count}(w_{i-k+1}^i)}{\text{count}(w_{i-k+1}^{i-1})} & \text{if } \text{count}(w_{i-k+1}^i) > 0 \\ 0.4S(w_i | w_{i-k+2}^{i-1}) & \text{otherwise} \end{cases}$$

$$S(w_i) = \frac{\text{count}(w_i)}{N}$$

N-gram Smoothing Summary

Add-1 smoothing:

- OK for text categorization, not for language modeling

The most commonly used method:

- Extended Interpolated Kneser-Ney

For very large N-grams like the Web:

- Stupid backoff

Advanced Language Modeling

Discriminative models:

- choose n-gram weights to improve a task, not to fit the training set

Parsing-based models

Caching Models

- Recently used words are more likely to appear

$$P_{CACHE}(w | history) = \lambda P(w_i | w_{i-2}w_{i-1}) + (1 - \lambda) \frac{c(w \in history)}{|history|}$$

- These turned out to perform very poorly for speech recognition (why?)

Language Modeling

Interpolation, Backoff, and Web-Scale LMs

Language Modeling

Advanced:
Kneser-Ney Smoothing

Absolute discounting: just subtract a little from each count

Suppose we wanted to subtract a little from a count of 4 to save probability mass for the zeros

How much to subtract ?

Church and Gale (1991)'s clever idea

Divide up 22 million words of AP Newswire

- Training and held-out set
- for each bigram in the training set
- see the actual count in the held-out set!

It sure looks like $c^* = (c - .75)$

Bigram count in training	Bigram count in heldout set
0	.0000270
1	0.448
2	1.25
3	2.24
4	3.23
5	4.21
6	5.23
7	6.21
8	7.21
9	8.26

Absolute Discounting Interpolation

Save ourselves some time and just subtract 0.75 (or some d)!

$$P_{\text{AbsoluteDiscounting}}(w_i | w_{i-1}) = \frac{c(w_{i-1}, w_i) - d}{c(w_{i-1})} + \lambda(\overset{\swarrow}{w_{i-1}}) P(\overset{\nwarrow}{w})$$

discounted bigram Interpolation weight

unigram

- (Maybe keeping a couple extra values of d for counts 1 and 2)

But should we really just use the regular unigram $P(w)$?

Kneser-Ney Smoothing I

Better estimate for probabilities of lower-order unigrams!

- Shannon game: *I can't see without my reading _____* *Kongses* ?
- “Kong” turns out to be more common than “glasses”
- ... but “Kong” always follows “Hong”

The unigram is useful exactly when we haven’t seen this bigram!

Instead of $P(w)$: “How likely is w ”

$P_{\text{continuation}}(w)$: “How likely is w to appear as a novel continuation?”

- For each word, count the number of bigram types it completes
- Every bigram type was a novel continuation the first time it was seen

$$P_{\text{CONTINUATION}}(w) \propto |\{w_{i-1} : c(w_{i-1}, w) > 0\}|$$

Kneser-Ney Smoothing II

How many times does w appear as a novel continuation:

$$P_{CONTINUATION}(w) \propto |\{w_{i-1} : c(w_{i-1}, w) > 0\}|$$

Normalized by the total number of word bigram types

$$\left| \{(w_{j-1}, w_j) : c(w_{j-1}, w_j) > 0\} \right|$$

$$P_{CONTINUATION}(w) = \frac{\left| \{w_{i-1} : c(w_{i-1}, w) > 0\} \right|}{\left| \{(w_{j-1}, w_j) : c(w_{j-1}, w_j) > 0\} \right|}$$

Kneser-Ney Smoothing III

Alternative metaphor: The number of # of word types seen to precede w

$$|\{w_{i-1} : c(w_{i-1}, w) > 0\}|$$

normalized by the # of words preceding all words:

$$P_{CONTINUATION}(w) = \frac{|\{w_{i-1} : c(w_{i-1}, w) > 0\}|}{\sum_{w'} |\{w'_{i-1} : c(w'_{i-1}, w') > 0\}|}$$

A frequent word (Kong) occurring in only one context (Hong) will have a low continuation probability

Kneser-Ney Smoothing IV

$$P_{KN}(w_i | w_{i-1}) = \frac{\max(c(w_{i-1}, w_i) - d, 0)}{c(w_{i-1})} + \lambda(w_{i-1}) P_{CONTINUATION}(w_i)$$

λ is a normalizing constant; the probability mass we've discounted

$$\lambda(w_{i-1}) = \frac{d}{c(w_{i-1})} |\{w : c(w_{i-1}, w) > 0\}|$$

the normalized discount

The number of word types that can follow w_{i-1}
= # of word types we discounted
= # of times we applied normalized discount

Kneser-Ney Smoothing: Recursive formulation

$$P_{KN}(w_i | w_{i-n+1}^{i-1}) = \frac{\max(c_{KN}(w_{i-n+1}^i) - d, 0)}{c_{KN}(w_{i-n+1}^{i-1})} + \lambda(w_{i-n+1}^{i-1}) P_{KN}(w_i | w_{i-n+2}^{i-1})$$

$$c_{KN}(\bullet) = \begin{cases} count(\bullet) & \text{for the highest order} \\ continuationcount(\bullet) & \text{for lower order} \end{cases}$$

Continuation count = Number of unique single word contexts for \bullet

Language Modeling

Advanced:
Kneser-Ney Smoothing

Spelling Correction: Edit Distance

Pawan Goyal

CSE, IITKGP

Week 2: Lecture 1

Spelling Correction

NPTEL

Spelling Correction

I am writing this email on behaf of ...

NPTEL

Spelling Correction

I am writing this email on behaf of ...

The user typed ‘behaf’.

Which are some close words?

Spelling Correction

I am writing this email on behaf of ...

The user typed ‘behaf’.

Which are some close words?

- behalf
- behave
-

Spelling Correction

I am writing this email on behaf of ...

The user typed ‘behaf’.

Which are some close words?

- behalf
- behave
-

Isolated word error correction

- Pick the one that is closest to ‘behaf’

Spelling Correction

I am writing this email on behaf of ...

The user typed ‘behaf’.

Which are some close words?

- behalf
- behave
-

Isolated word error correction

- Pick the one that is closest to ‘behaf’
- How to define ‘closest’?

Spelling Correction

I am writing this email on behaf of ...

The user typed ‘behaf’.

Which are some close words?

- behalf
- behave
-

Isolated word error correction

- Pick the one that is closest to ‘behaf’
- How to define ‘closest’?
- Need a **distance metric**

Spelling Correction

I am writing this email on behaf of ...

The user typed 'behaf'.

Which are some close words?

- behalf
- behave
-

Isolated word error correction

- Pick the one that is closest to 'behaf'
- How to define 'closest'?
- Need a **distance metric**
- The simplest metric: **edit distance**

Edit Distance

- The minimum edit distance between two strings

Edit Distance

- The minimum edit distance between two strings
- Is the minimum number of editing operations

Edit Distance

- The minimum edit distance between two strings
- Is the minimum number of editing operations
 - ▶ Insertion
 - ▶ Deletion
 - ▶ Substitution

Minimum Edit Distance

Example

Edit distance from ‘intention’ to ‘execution’

Minimum Edit Distance

Example

Edit distance from ‘intention’ to ‘execution’

I	N	T	E	*	N	T	I	O	N
*	E	X	E	C	U	T	I	O	N

Minimum Edit Distance

I	N	T	E	*	N	T	I	O	N
*	E	X	E	C	U	T	I	O	N
d	s	s		i	s				

Minimum Edit Distance

I	N	T	E	*	N	T	I	O	N
*	E	X	E	C	U	T	I	O	N
d	s	s		i	s				

- If each operation has a cost of 1 (Levenshtein)
 - ▶ Distance between these is 5

Minimum Edit Distance

I	N	T	E	*	N	T	I	O	N
*	E	X	E	C	U	T	I	O	N
d	s	s		i	s				

- If each operation has a cost of 1 (Levenshtein)
 - ▶ Distance between these is 5
- If substitution costs 2 (alternate version)
 - ▶ Distance between these is 8

How to find the Minimum Edit Distance?

NPTEL

How to find the Minimum Edit Distance?

Searching for a path (sequence of edits) from the *start string* to the *final string*:

How to find the Minimum Edit Distance?

Searching for a path (sequence of edits) from the *start string* to the *final string*:

- **Initial state:** the word we are transforming

How to find the Minimum Edit Distance?

Searching for a path (sequence of edits) from the *start string* to the *final string*:

- **Initial state:** the word we are transforming
- **Operators:** insert, delete, substitute

How to find the Minimum Edit Distance?

Searching for a path (sequence of edits) from the *start string* to the *final string*:

- **Initial state:** the word we are transforming
- **Operators:** insert, delete, substitute
- **Goal state:** the word we are trying to get to

How to find the Minimum Edit Distance?

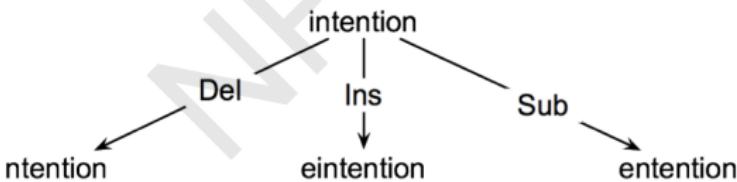
Searching for a path (sequence of edits) from the *start string* to the *final string*:

- **Initial state:** the word we are transforming
- **Operators:** insert, delete, substitute
- **Goal state:** the word we are trying to get to
- **Path cost:** what we want to minimize: the number of edits

How to find the Minimum Edit Distance?

Searching for a path (sequence of edits) from the *start string* to the *final string*:

- **Initial state:** the word we are transforming
- **Operators:** insert, delete, substitute
- **Goal state:** the word we are trying to get to
- **Path cost:** what we want to minimize: the number of edits



Minimum Edit as Search

How to navigate?

- The space of all edit sequences is huge

Minimum Edit as Search

How to navigate?

- The space of all edit sequences is huge
- Lot of distinct paths end up at the same state

Minimum Edit as Search

How to navigate?

- The space of all edit sequences is huge
- Lot of distinct paths end up at the same state
- Don't have to keep track of all of them

Minimum Edit as Search

How to navigate?

- The space of all edit sequences is huge
- Lot of distinct paths end up at the same state
- Don't have to keep track of all of them
- Keep track of the shortest path to each state

Defining Minimum Edit Distance Matrix

For two strings

- X of length n
- Y of length m

Defining Minimum Edit Distance Matrix

For two strings

- X of length n
- Y of length m

We define $D(i,j)$

- the edit distance between $X[1..i]$ and $Y[1..j]$
- i.e., the first i characters of X and the first j characters of Y

Defining Minimum Edit Distance Matrix

For two strings

- X of length n
- Y of length m

We define $D(i,j)$

- the edit distance between $X[1..i]$ and $Y[1..j]$
- i.e., the first i characters of X and the first j characters of Y

Thus, the edit distance between X and Y is $D(n,m)$

Computing Minimum Edit Distance

Dynamic Programming

- A tabular computation of $D(n, m)$

Computing Minimum Edit Distance

Dynamic Programming

- A tabular computation of $D(n, m)$
- Solving problems by combining solutions to subproblems

Computing Minimum Edit Distance

Dynamic Programming

- A tabular computation of $D(n, m)$
- Solving problems by combining solutions to subproblems
- Bottom-up

Computing Minimum Edit Distance

Dynamic Programming

- A tabular computation of $D(n,m)$
- Solving problems by combining solutions to subproblems
- Bottom-up
 - ▶ Compute $D(i,j)$ for small i,j

Computing Minimum Edit Distance

Dynamic Programming

- A tabular computation of $D(n,m)$
- Solving problems by combining solutions to subproblems
- Bottom-up
 - ▶ Compute $D(i,j)$ for small i,j
 - ▶ Compute larger $D(i,j)$ based on previously computed smaller values

Computing Minimum Edit Distance

Dynamic Programming

- A tabular computation of $D(n,m)$
- Solving problems by combining solutions to subproblems
- Bottom-up
 - ▶ Compute $D(i,j)$ for small i,j
 - ▶ Compute larger $D(i,j)$ based on previously computed smaller values
 - ▶ Compute $D(i,j)$ for all i and j till you get to $D(n,m)$

Dynamic Programming Algorithm

Initialization

$$D(i, 0) = i$$

$$D(0, j) = j$$

Recurrence Relation:

For each $i = 1 \dots M$

For each $j = 1 \dots N$

$$D(i, j) = \min \begin{cases} D(i-1, j) + 1 \\ D(i, j-1) + 1 \\ D(i-1, j-1) + 2; & \text{if } X(i) \neq Y(j) \\ 0; & \text{if } X(i) = Y(j) \end{cases}$$

Termination:

$D(N, M)$ is distance

The Edit Distance Table

N	9										
O	8										
I	7										
T	6										
N	5										
E	4										
T	3										
N	2										
I	1										
#	0	1	2	3	4	5	6	7	8	9	
	#	E	X	E	C	U	T	I	O	N	

The Edit Distance Table

N	9										
O	8										
I	7										
T	6										
N	5										
E	4										
T	3										
N	2										
I	1										
#	0	1	2	3	4	5	6	7	8	9	
	#	E	X	E	C	U	T	I	O	N	

$$D(i,j) = \min \begin{cases} D(i-1,j) + 1 \\ D(i,j-1) + 1 \\ D(i-1,j-1) + \begin{cases} 2; & \text{if } S_1(i) \neq S_2(j) \\ 0; & \text{if } S_1(i) = S_2(j) \end{cases} \end{cases}$$

The Edit Distance Table

N	9	8	9	10	11	12	11	10	9	8
O	8	7	8	9	10	11	10	9	8	9
I	7	6	7	8	9	10	9	8	9	10
T	6	5	6	7	8	9	8	9	10	11
N	5	4	5	6	7	8	9	10	11	10
E	4	3	4	5	6	7	8	9	10	9
T	3	4	5	6	7	8	7	8	9	8
N	2	3	4	5	6	7	8	7	8	7
I	1	2	3	4	5	6	7	6	7	8
#	0	1	2	3	4	5	6	7	8	9
	#	E	X	E	C	U	T	I	O	N

Computing Alignments

NPTEL

Computing Alignments

- Computing edit distance may not be sufficient for some applications

Computing Alignments

- Computing edit distance may not be sufficient for some applications
 - ▶ We often need to align characters of the two strings to each other

Computing Alignments

- Computing edit distance may not be sufficient for some applications
 - ▶ We often need to align characters of the two strings to each other
 - We do this by keeping a “backtrace”

Computing Alignments

- Computing edit distance may not be sufficient for some applications
 - ▶ We often need to align characters of the two strings to each other
- We do this by keeping a “backtrace”
- Every time we enter a cell, remember where we came from

Computing Alignments

- Computing edit distance may not be sufficient for some applications
 - ▶ We often need to align characters of the two strings to each other
- We do this by keeping a “backtrace”
- Every time we enter a cell, remember where we came from
- When we reach the end,

Computing Alignments

- Computing edit distance may not be sufficient for some applications
 - ▶ We often need to align characters of the two strings to each other
- We do this by keeping a “backtrace”
- Every time we enter a cell, remember where we came from
- When we reach the end,
 - ▶ Trace back the path from the upper right corner to read off the alignment

The Edit Distance Table

N	9									
O	8									
I	7									
T	6									
N	5									
E	4									
T	3									
N	2									
I	1									
#	0	1	2	3	4	5	6	7	8	9
	#	E	X	E	C	U	T	I	O	N

$$D(i,j) = \min \begin{cases} D(i-1,j) + 1 \\ D(i,j-1) + 1 \\ D(i-1,j-1) + \begin{cases} 2; & \text{if } S_1(i) \neq S_2(j) \\ 0; & \text{if } S_1(i) = S_2(j) \end{cases} \end{cases}$$

The Edit Distance Table

N	9									
O	8									
I	7									
T	6									
N	5									
E	4	3	4							
T	3	4	5							
N	2	3	4							
I	1	2	3							
#	0	1	2	3	4	5	6	7	8	9
	#	E	X	E	C	U	T	I	O	N

$$D(i,j) = \min \begin{cases} D(i-1,j) + 1 \\ D(i,j-1) + 1 \\ D(i-1,j-1) + \begin{cases} 2; & \text{if } S_1(i) \neq S_2(j) \\ 0; & \text{if } S_1(i) = S_2(j) \end{cases} \end{cases}$$

Minimum Edit with Backtrace

n	9	↓ 8	↙ ↘ 9	↙ ↘ 10	↙ ↘ 11	↙ ↘ 12	↓ 11	↓ 10	↓ 9	↙ 8	
o	8	↓ 7	↙ ↘ 8	↙ ↘ 9	↙ ↘ 10	↙ ↘ 11	↓ 10	↓ 9	↙ 8	← 9	
i	7	↓ 6	↙ ↘ 7	↙ ↘ 8	↙ ↘ 9	↙ ↘ 10	↓ 9	↙ 8	← 9	← 10	
t	6	↓ 5	↙ ↘ 6	↙ ↘ 7	↙ ↘ 8	↙ ↘ 9	↙ 8	← 9	← 10	← 11	
n	5	↓ 4	↙ ↘ 5	↙ ↘ 6	↙ ↘ 7	↙ ↘ 8	↙ ↘ 9	↙ ↘ 10	↙ ↘ 11	↙ ↘ 10	
e	4	↙ 3	← 4	↙ ↘ 5	← 6	← 7	← 8	↙ ↘ 9	↙ ↘ 10	↓ 9	
t	3	↙ ↘ 4	↙ ↘ 5	↙ ↘ 6	↙ ↘ 7	↙ ↘ 8	↙ 7	← 8	↙ ↘ 9	↓ 8	
n	2	↙ ↘ 3	↙ ↘ 4	↙ ↘ 5	↙ ↘ 6	↙ ↘ 7	↙ ↘ 8	↓ 7	↙ ↘ 8	↙ 7	
i	1	↙ ↘ 2	↙ ↘ 3	↙ ↘ 4	↙ ↘ 5	↙ ↘ 6	↙ ↘ 7	↙ 6	← 7	← 8	
#	0	1	2	3	4	5	6	7	8	9	
	#	e	x	e	c	u	t	i	o	n	

Adding Backtrace to Minimum Edit

Base conditions:

$$D(i, 0) = i$$

$$D(0, j) = j$$

Termination:

$D(N, M)$ is distance

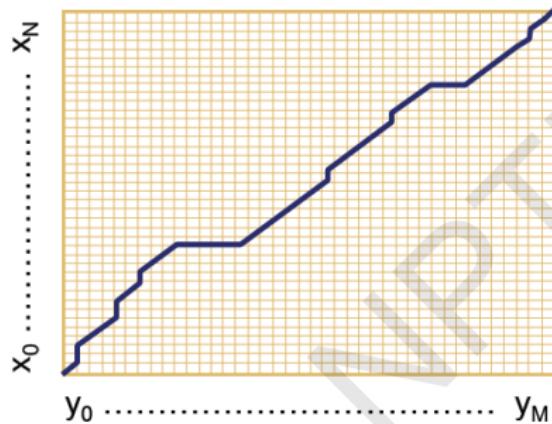
Recurrence Relation:

For each $i = 1 \dots M$

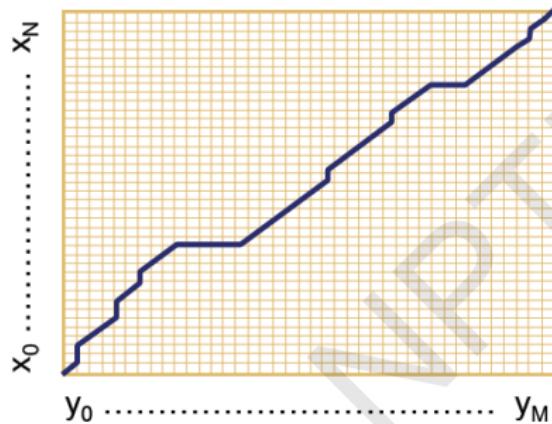
For each $j = 1 \dots N$

$$D(i, j) = \min \begin{cases} D(i-1, j) + 1 & \text{deletion} \\ D(i, j-1) + 1 & \text{insertion} \\ D(i-1, j-1) + 2; & \begin{cases} \text{if } X(i) \neq Y(j) \\ 0; \quad \text{if } X(i) = Y(j) \end{cases} \\ \text{substitution} \end{cases}$$
$$\text{ptr}(i, j) = \begin{cases} \text{LEFT} & \text{insertion} \\ \text{DOWN} & \text{deletion} \\ \text{DIAG} & \text{substitution} \end{cases}$$

The distance matrix

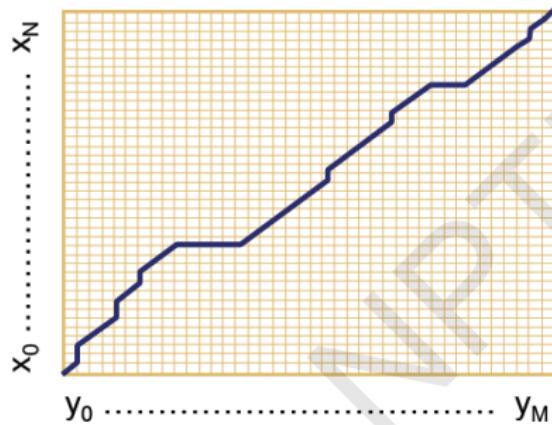


The distance matrix



Every non-decreasing path from (0,0) to (M,N) corresponds to an alignment of two sequences.

The distance matrix



Every non-decreasing path from $(0,0)$ to (M,N) corresponds to an alignment of two sequences.

An optimal alignment is composed of optimal sub-alignments.

Result of Backtrace

I	N	T	E	*	N	T	I	O	N
*	E	X	E	C	U	T	I	O	N

Performance

Time

NPTEL

Performance

Time

$O(nm)$

Space

Performance

Time

$O(nm)$

Space

$O(nm)$

Backtrace

Performance

Time

$O(nm)$

Space

$O(nm)$

Backtrace

$O(n + m)$

Weighted Edit Distance, Other variations

Pawan Goyal

CSE, IITKGP

Week 2: Lecture 2

Weighted Edit Distance

Why to add weights to the computation?

- Some letters are more likely to be mistyped.

Confusion Matrix for Spelling Errors

sub[X, Y] = Substitution of X (incorrect) for Y (correct)
Y (correct)

X	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
a	0	0	7	1	342	0	0	2	118	0	1	0	0	3	76	0	0	1	35	9	9	0	1	0	5	0
b	0	0	9	9	2	2	3	1	0	0	0	5	11	5	0	10	0	0	2	1	0	0	8	0	0	0
c	6	5	0	16	0	9	5	0	0	0	1	0	7	9	1	10	2	5	39	40	1	3	7	1	1	0
d	1	10	13	0	12	0	5	5	0	0	2	3	7	3	0	1	0	43	30	22	0	0	4	0	2	0
e	388	0	3	11	0	2	2	0	89	0	0	3	0	5	93	0	0	14	12	6	15	0	1	0	18	0
f	0	15	0	3	1	0	5	2	0	0	0	3	4	1	0	0	0	6	4	12	0	0	2	0	0	0
g	4	1	11	11	9	2	0	0	0	1	1	3	0	0	2	1	3	5	13	21	0	0	1	0	3	0
h	1	8	0	3	0	0	0	0	0	0	2	0	12	14	2	3	0	3	1	11	0	0	2	0	0	0
i	103	0	0	0	146	0	1	0	0	0	0	6	0	0	49	0	0	0	2	1	47	0	2	1	15	0
j	0	1	1	9	0	0	1	0	0	0	0	2	1	0	0	0	0	0	5	0	0	0	0	0	0	0
k	1	2	8	4	1	1	2	5	0	0	0	0	5	0	2	0	0	0	6	0	0	0	4	0	0	3
l	2	10	1	4	0	4	5	6	13	0	1	0	0	14	2	5	0	11	10	2	0	0	0	0	0	0
m	1	3	7	8	0	2	0	6	0	0	4	4	0	180	0	6	0	0	9	15	13	3	2	2	3	0
n	2	7	6	5	3	0	1	19	1	0	4	35	78	0	0	7	0	28	5	7	0	0	1	2	0	2
o	91	1	1	3	116	0	0	0	25	0	2	0	0	0	0	14	0	2	4	14	39	0	0	0	18	0
p	0	11	1	2	0	6	5	0	2	9	0	2	7	6	15	0	0	1	3	6	0	4	1	0	0	0
q	0	0	1	0	0	0	27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
r	0	14	0	30	12	2	2	8	2	0	5	8	4	20	1	14	0	0	12	22	4	0	0	1	0	0
s	11	8	27	33	35	4	0	1	0	1	0	27	0	6	1	7	0	14	0	15	0	0	5	3	20	1
t	3	4	9	42	7	5	19	5	0	1	0	14	9	5	5	6	0	11	37	0	0	2	19	0	7	6
u	20	0	0	0	44	0	0	0	64	0	0	0	0	2	43	0	0	4	0	0	0	0	2	0	8	0
v	0	0	7	0	0	3	0	0	0	0	0	1	0	0	1	0	0	8	3	0	0	0	0	0	0	0
w	2	2	1	0	1	0	0	2	0	0	1	0	0	0	0	7	0	6	3	3	1	0	0	0	0	0
x	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	9	0	0	0	0	0	0	0
y	0	0	2	0	15	0	1	7	15	0	0	0	2	0	6	1	0	7	36	8	5	0	0	1	0	0
z	0	0	0	7	0	0	0	0	0	0	0	7	5	0	0	0	0	2	21	3	0	0	0	0	3	0

Keyboard Design



Weighted Minimum Edit Distance

Initialization:

$$D(0,0) = 0$$

$$D(i,0) = D(i-1,0) + \text{del}[x(i)]; \quad 1 < i \leq N$$

$$D(0,j) = D(0,j-1) + \text{ins}[y(j)]; \quad 1 < j \leq M$$

Recurrence Relation:

$$D(i,j) = \min \begin{cases} D(i-1,j) + \text{del}[x(i)] \\ D(i,j-1) + \text{ins}[y(j)] \\ D(i-1,j-1) + \text{sub}[x(i), y(j)] \end{cases}$$

Termination:

$D(N,M)$ is distance

How to modify the algorithm with transpose?

Transpose

- $\text{transpose}(x, y) = (y, x)$
- Also known as metathesis

How to modify the algorithm with transpose?

Transpose

- $\text{transpose}(x, y) = (y, x)$
- Also known as metathesis

Modification to the dynamic programmic algorithm

$$D[i][j] = \min \begin{cases} D(i-1, j) + 1 & (\text{deletion}) \\ D(i, j-1) + 1 & (\text{insertion}) \\ D(i-1, j-1) + \begin{cases} 1 & \text{if } (x[i] \neq y[j]) \text{ (substitution)} \\ 0 & \text{otherwise} \end{cases} & \\ D(i-2, j-2) + 1 & (x[i] = y[j-1] \text{ and } x[i-1] = y[j]) \\ & (\text{transposition}) \end{cases}$$

How to find dictionary entries with smallest edit distance?

NPTEL

How to find dictionary entries with smallest edit distance?

Naïve Method

Compute edit ditance from the query term to each dictionary term – an exhaustive search

How to find dictionary entries with smallest edit distance?

Naïve Method

Compute edit distance from the query term to each dictionary term – an exhaustive search

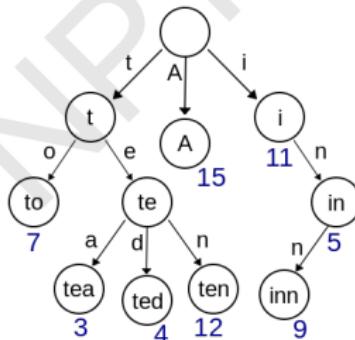
Can be made efficient if we do it over a trie structure

How to find dictionary entries with smallest edit distance?

Naïve Method

Compute edit distance from the query term to each dictionary term – an exhaustive search

Can be made efficient if we do it over a trie structure



How to find dictionary entries with smallest edit distance?

NPTEL

How to find dictionary entries with smallest edit distance?

- Generate all possible terms with an edit distance ≤ 2 (deletion + transpose + substitution + insertion) from the query term and search them in the dictionary.

How to find dictionary entries with smallest edit distance?

- Generate all possible terms with an edit distance ≤ 2 (deletion + transpose + substitution + insertion) from the query term and search them in the dictionary.
- For a word of length 9, alphabet of size 36, this will lead to 114,324 terms to search for

How to find dictionary entries with smallest edit distance?

- Generate all possible terms with an edit distance ≤ 2 (deletion + transpose + substitution + insertion) from the query term and search them in the dictionary.
- For a word of length 9, alphabet of size 36, this will lead to 114,324 terms to search for
- For Chinese alphabet size is 70,000 (Unicode Han Characters)

How to find dictionary entries with smallest edit distance?

Symmetric Delete Spelling Correction

- Generate terms with an edit distance ≤ 2 (deletes) from each dictionary term (offline)
- Generate terms with an edit distance ≤ 2 (deletes) from the input terms and search in dictionary

How to find dictionary entries with smallest edit distance?

Symmetric Delete Spelling Correction

- Generate terms with an edit distance ≤ 2 (deletes) from each dictionary term (offline)
- Generate terms with an edit distance ≤ 2 (deletes) from the input terms and search in dictionary

Number of deletes within edit distance ≤ 2 for a word of length 9 will be 45

How to find dictionary entries with smallest edit distance?

Symmetric Delete Spelling Correction

- Generate terms with an edit distance ≤ 2 (deletes) from each dictionary term (offline)
- Generate terms with an edit distance ≤ 2 (deletes) from the input terms and search in dictionary

Number of deletes within edit distance ≤ 2 for a word of length 9 will be 45

A further check is required to remove the false positives

Spelling Correction

NPTEL

Spelling Correction

Types of spelling errors: Non-word Errors

- behaf → behalf

Spelling Correction

Types of spelling errors: Non-word Errors

- behaf → behalf

Types of spelling errors: Real-word Errors

- **Typographical errors:** three → there
- **Cognitive errors (homophones):** piece → peace, too → two

Non-word spelling errors

Non-word spelling error detection

- Any word not in a dictionary is an error
- The larger the dictionary the better

Non-word spelling errors

Non-word spelling error detection

- Any word not in a dictionary is an error
- The larger the dictionary the better

Non-word spelling error correction

- Generate candidates: real words that are similar to the error word
- Choose the best one:
 - ▶ Shortest weighted edit distance
 - ▶ Highest noisy channel probability

Real word spelling errors

For each word w , generate candidate set

- Find candidate words with similar pronunciations
- Find candidate words with similar spelling
- Include w in candidate set

Real word spelling errors

For each word w , generate candidate set

- Find candidate words with similar pronunciations
- Find candidate words with similar spelling
- Include w in candidate set

Choosing best candidate

- Noisy Channel

Noisy Channel Model for Spelling Correction

Pawan Goyal

CSE, IITKGP

Week 2: Lecture 3

Noisy Channel

We see an observation x of the misspelled word

Find the correct word w

$$\hat{w} = \arg \max_{w \in V} P(w|x)$$

Noisy Channel

We see an observation x of the misspelled word

Find the correct word w

$$\begin{aligned}\hat{w} &= \arg \max_{w \in V} P(w|x) \\ &= \arg \max_{w \in V} \frac{P(x|w)P(w)}{P(x)}\end{aligned}$$

Noisy Channel

We see an observation x of the misspelled word

Find the correct word w

$$\begin{aligned}\hat{w} &= \arg \max_{w \in V} P(w|x) \\ &= \arg \max_{w \in V} \frac{P(x|w)P(w)}{P(x)} \\ &= \arg \max_{w \in V} P(x|w)P(w)\end{aligned}$$

*Non-word spelling error: *acress**

Words with similar spelling

Small edit distance to error

Words with similar pronunciation

Small edit distance of pronunciation to error

*Non-word spelling error: *acress**

Words with similar spelling

Small edit distance to error

Words with similar pronunciation

Small edit distance of pronunciation to error

Damerau-Levenshtein edit distance

Minimum edit distance, where edits are:

*Non-word spelling error: *acress**

Words with similar spelling

Small edit distance to error

Words with similar pronunciation

Small edit distance of pronunciation to error

Damerau-Levenshtein edit distance

Minimum edit distance, where edits are:

Insertion, Deletion, Substitution,

*Non-word spelling error: *acress**

Words with similar spelling

Small edit distance to error

Words with similar pronunciation

Small edit distance of pronunciation to error

Damerau-Levenshtein edit distance

Minimum edit distance, where edits are:

Insertion, Deletion, Substitution,

Transposition of two adjacent letters

Words within edit distance 1 of acress

Error	Candidate Correction	Correct Letter	Error Letter	Type
acress	actress	t	-	deletion
acress	cress	-	a	insertion
acress	caress	ca	ac	transposition
acress	access	c	r	substitution
acress	across	o	e	substitution
acress	acres	-	s	insertion
acress	acres	-	s	insertion

Candidate generation

- 80% of errors are within edit distance 1
- Almost all errors within edit distance 2

Candidate generation

- 80% of errors are within edit distance 1
- Almost all errors within edit distance 2

Allow deletion of space or hyphen

- thisidea → this idea
- inlaw → in-law

Computing error probability: confusion matrix

- $\text{del}[x,y]$: count (xy typed as x)
- $\text{ins}[x,y]$: count (x typed as xy)
- $\text{sub}[x,y]$: count (x typed as y)
- $\text{trans}[x,y]$: count(xy typed as yx)

Computing error probability: confusion matrix

- $\text{del}[x,y]$: count (xy typed as x)
- $\text{ins}[x,y]$: count (x typed as xy)
- $\text{sub}[x,y]$: count (x typed as y)
- $\text{trans}[x,y]$: count(xy typed as yx)

Insertion and deletion are conditioned on previous character

Channel model

$$P(x|w) = \begin{cases} \frac{\text{del}[w_{i-1}, w_i]}{\text{count}[w_{i-1} w_i]}, & \text{if deletion} \\ \frac{\text{ins}[w_{i-1}, x_i]}{\text{count}[w_{i-1}]}, & \text{if insertion} \\ \frac{\text{sub}[x_i, w_i]}{\text{count}[w_i]}, & \text{if substitution} \\ \frac{\text{trans}[w_i, w_{i+1}]}{\text{count}[w_i w_{i+1}]}, & \text{if transposition} \end{cases}$$

Channel model for across

Candidate Correction	Correct Letter	Error Letter	x w	P(x word)
actress	t	-	c ct	.000117
cress	-	a	a #	.00000144
caress	ca	ac	ac ca	.00000164
access	c	r	r c	.000000209
across	o	e	e o	.0000093
acres	-	s	es e	.0000321
acres	-	s	ss s	.0000342

Noisy channel probability for *acress*

Candidate Correction	Correct Letter	Error Letter	x w	P(x word)	P(word)	$10^9 * P(x w)P(w)$
actress	t	-	c ct	.000117	.0000231	2.7
cress	-	a	a #	.00000144	.000000544	.00078
caress	ca	ac	ac ca	.00000164	.00000170	.0028
access	c	r	r c	.000000209	.0000916	.019
across	o	e	e o	.0000093	.000299	2.8
acres	-	s	es e	.0000321	.0000318	1.0
acres	-	s	ss s	.0000342	.0000318	1.0

Using a bigram language model

- “ ... versatile across whose ... ”

Using a bigram language model

- “ ... versatile across whose ... ”
- Counts from the Corpus of Contemporary American English with add-1 smoothing

Using a bigram language model

- “ ... versatile across whose ... ”
- Counts from the Corpus of Contemporary American English with add-1 smoothing
- $P(\text{actress}|\text{versatile}) = 0.000021$, $P(\text{across}|\text{versatile}) = 0.000021$

Using a bigram language model

- “ ... versatile across whose ... ”
- Counts from the Corpus of Contemporary American English with add-1 smoothing
- $P(\text{actress}|\text{versatile}) = 0.000021$, $P(\text{across}|\text{versatile}) = 0.000021$
- $P(\text{whose}|\text{actress}) = 0.0010$, $P(\text{whose}|\text{across}) = 0.000006$

Using a bigram language model

- “ ... versatile across whose ... ”
- Counts from the Corpus of Contemporary American English with add-1 smoothing
- $P(\text{actress}|\text{versatile}) = 0.000021$, $P(\text{across}|\text{versatile}) = 0.000021$
- $P(\text{whose}|\text{actress}) = 0.0010$, $P(\text{whose}|\text{across}) = 0.000006$
- $P(\text{"versatile actress whose"}) = 0.000021 * 0.0010 = 210 \times 10^{-10}$

Using a bigram language model

- “ ... versatile across whose ... ”
- Counts from the Corpus of Contemporary American English with add-1 smoothing
- $P(\text{actress}|\text{versatile}) = 0.000021, P(\text{across}|\text{versatile}) = 0.000021$
- $P(\text{whose}|\text{actress}) = 0.0010, P(\text{whose}|\text{across}) = 0.000006$
- $P(\text{"versatile actress whose"}) = 0.000021 * 0.0010 = 210 \times 10^{-10}$
- $P(\text{"versatile across whose"}) = 0.000021 * 0.000006 = 1 \times 10^{-10}$

Real-word spelling errors

- The study was conducted mainly **be** John Black
- The design **an** construction of the system ...

Real-word spelling errors

- The study was conducted mainly **be** John Black
- The design **an** construction of the system ...

25-40% of spelling errors are real words

Noisy channel for real-word spell correction

Given a sentence $X = w_1, w_2, w_3 \dots, w_n$

- Candidate (w_1) = $\{w_1, w'_1, w''_1, w'''_1, \dots\}$
- Candidate (w_2) = $\{w_2, w'_2, w''_2, w'''_2, \dots\}$
- Candidate (w_3) = $\{w_3, w'_3, w''_3, w'''_3, \dots\}$

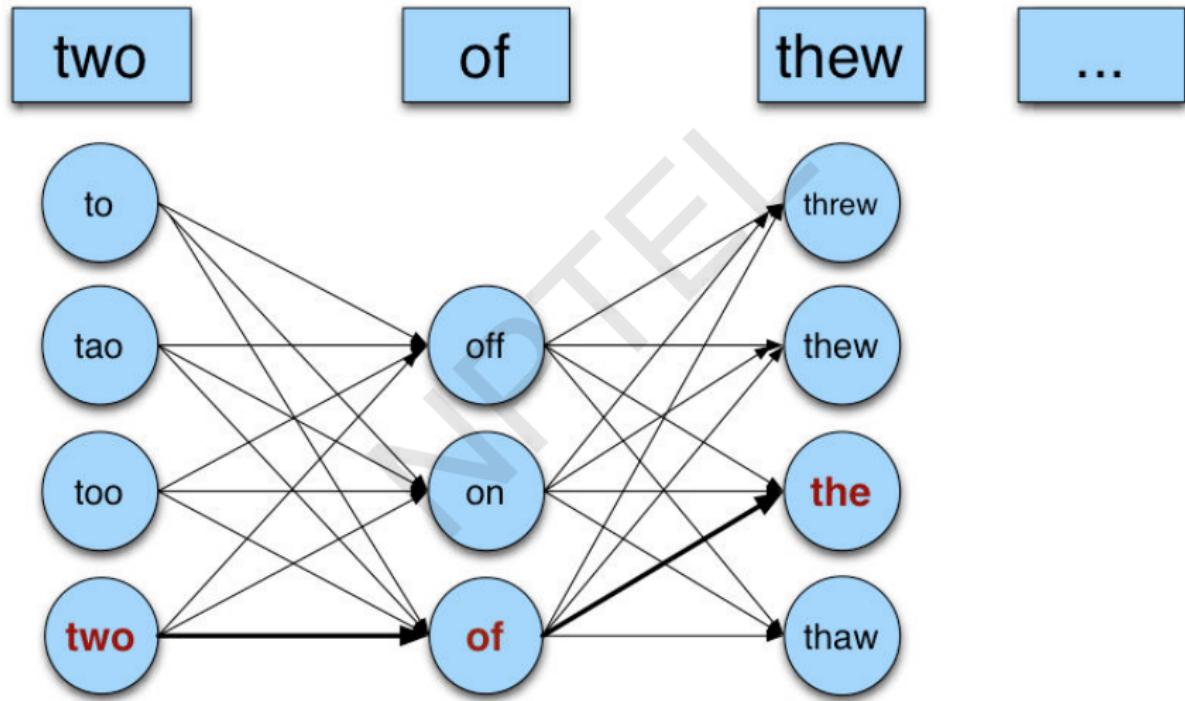
Noisy channel for real-word spell correction

Given a sentence $X = w_1, w_2, w_3 \dots, w_n$

- Candidate (w_1) = $\{w_1, w'_1, w''_1, w'''_1, \dots\}$
- Candidate (w_2) = $\{w_2, w'_2, w''_2, w'''_2, \dots\}$
- Candidate (w_3) = $\{w_3, w'_3, w''_3, w'''_3, \dots\}$

Choose the sequence W that maximizes $P(W|X)$

Noisy channel for real-world spell correction



Simplification: One error per sentence

Choose among all possible sentences with one word replaced

two of thew

- w_1, w''_2, w_3 two **off** thew
- w_1, w_2, w'_3 two of **the**
- w'''_1, w_2, w_3 **too** of thew

Simplification: One error per sentence

Choose among all possible sentences with one word replaced

two of thew

- w_1, w''_2, w_3 two **off** thew
- w_1, w_2, w'_3 two of **the**
- w'''_1, w_2, w_3 **too** of thew

Choose the sequence W that maximizes $P(W|X)$

Getting the probability values

Noisy Channel

$$\hat{W} = \arg \max_{W \in S} P(W|X)$$

where X is the observed sentence and S is the set of all the possible sequences from the candidate set

Getting the probability values

Noisy Channel

$$\hat{W} = \arg \max_{W \in S} P(W|X)$$

where X is the observed sentence and S is the set of all the possible sequences from the candidate set

$$= \arg \max_{W \in S} P(X|W)P(W)$$

Getting the probability values

Noisy Channel

$$\hat{W} = \arg \max_{W \in S} P(W|X)$$

where X is the observed sentence and S is the set of all the possible sequences from the candidate set

$$= \arg \max_{W \in S} P(X|W)P(W)$$

$P(X|W)$

- Same as for non-word spelling correction

Getting the probability values

Noisy Channel

$$\hat{W} = \arg \max_{W \in S} P(W|X)$$

where X is the observed sentence and S is the set of all the possible sequences from the candidate set

$$= \arg \max_{W \in S} P(X|W)P(W)$$

$P(X|W)$

- Same as for non-word spelling correction
- Also require probability for no error $P(w|w)$

Probability of no error

What is the probability for a correctly typed word? $P(\text{"the"}|\text{"the"})$

Probability of no error

What is the probability for a correctly typed word? $P(\text{"the"}|\text{"the"})$

It may depend on the source text under consideration

- 1 error in 10 words → 0.9
- 1 error in 100 words → 0.99

$P(W)$

Use Language Model

- Unigram
- Bigram
- ...

N-gram Language Models

Pawan Goyal

CSE, IITKGP

Week 2: Lecture 4

Context Sensitive Spelling Correction

NPTEL

Context Sensitive Spelling Correction

The office is about fifteen minuets from my house

Context Sensitive Spelling Correction

The office is about fifteen minuets from my house

min·u·et  noun \min-yə-'wet\

: a slow, graceful dance that was popular in the 17th and
18th centuries

: the music for a minuet

Context Sensitive Spelling Correction

The office is about fifteen minuets from my house

min·u·et  noun \min-yə-'wet\

: a slow, graceful dance that was popular in the 17th and 18th centuries

: the music for a minuet

Use a Language Model

$P(\text{about fifteen minutes from}) > P(\text{about fifteen minuets from})$

Probabilistic Language Models: Applications

Speech Recognition

- $P(I \text{ saw a van}) >> P(\text{eyes awe of an})$

Probabilistic Language Models: Applications

Speech Recognition

- $P(\text{I saw a van}) >> P(\text{eyes awe of an})$

Machine Translation

Which sentence is more plausible in the target language?

- $P(\text{high winds}) > P(\text{large winds})$

Probabilistic Language Models: Applications

Speech Recognition

- $P(I \text{ saw a van}) >> P(\text{eyes awe of an})$

Machine Translation

Which sentence is more plausible in the target language?

- $P(\textbf{high} \text{ winds}) > P(\textbf{large} \text{ winds})$

Other Applications

- Context Sensitive Spelling Correction
- Natural Language Generation
- ...

Completion Prediction

- Language model also supports predicting the completion of a sentence.
 - ▶ Please turn off your cell ...
 - ▶ Your program does not ...

Completion Prediction

- Language model also supports predicting the completion of a sentence.
 - ▶ Please turn off your cell ...
 - ▶ Your program does not ...
- *Predictive text input* systems can guess what you are typing and give choices on how to complete it.

Probabilistic Language Modeling

- **Goal:** Compute the probability of a sentence or sequence of words:

$$P(W) = P(w_1, w_2, w_3, \dots, w_n)$$

Probabilistic Language Modeling

- **Goal:** Compute the probability of a sentence or sequence of words:

$$P(W) = P(w_1, w_2, w_3, \dots, w_n)$$

- **Related Task:** probability of an upcoming word:

$$P(w_4 | w_1, w_2, w_3)$$

Probabilistic Language Modeling

- **Goal:** Compute the probability of a sentence or sequence of words:

$$P(W) = P(w_1, w_2, w_3, \dots, w_n)$$

- **Related Task:** probability of an upcoming word:

$$P(w_4 | w_1, w_2, w_3)$$

- A model that computes either of these is called a **language model**

$P(W)$

How to compute the joint probability

$P(\text{about, fifteen, minutes, from})$

$P(W)$

How to compute the joint probability

$P(\text{about, fifteen, minutes, from})$

Basic Idea

Rely on the Chain Rule of Probability

The Chain Rule

Conditional Probabilities

$$P(B|A) = \frac{P(A, B)}{P(A)}$$

The Chain Rule

Conditional Probabilities

$$P(B|A) = \frac{P(A, B)}{P(A)}$$

$$P(A, B) = P(A)P(B|A)$$

The Chain Rule

Conditional Probabilities

$$P(B|A) = \frac{P(A, B)}{P(A)}$$

$$P(A, B) = P(A)P(B|A)$$

More Variables

$$P(A, B, C, D) = P(A)P(B|A)P(C|A, B)P(D|A, B, C)$$

The Chain Rule

Conditional Probabilities

$$P(B|A) = \frac{P(A, B)}{P(A)}$$

$$P(A, B) = P(A)P(B|A)$$

More Variables

$$P(A, B, C, D) = P(A)P(B|A)P(C|A, B)P(D|A, B, C)$$

The Chain Rule in General

$$P(x_1, x_2, \dots, x_n) = P(x_1)P(x_2|x_1)P(x_3|x_1, x_2)\dots P(x_n|x_1, \dots, x_{n-1})$$

Probability of words in sentences

$$P(w_1 w_2 \dots w_n) = \prod_i P(w_i | w_1 w_2 \dots w_{i-1})$$

P("about fifteen minutes from") =

Probability of words in sentences

$$P(w_1 w_2 \dots w_n) = \prod_i P(w_i | w_1 w_2 \dots w_{i-1})$$

P("about fifteen minutes from") =

P(about) x P(fifteen | about) x P(minutes | about fifteen) x P(from | about fifteen minutes)

Estimating These Probability Values

Count and divide

$$P(\text{office} \mid \text{about fifteen minutes from}) = \frac{\text{Count}(\text{about fifteen minutes from office})}{\text{Count}(\text{about fifteen minutes from})}$$

Estimating These Probability Values

Count and divide

$$P(\text{office} \mid \text{about fifteen minutes from}) = \frac{\text{Count}(\text{about fifteen minutes from office})}{\text{Count}(\text{about fifteen minutes from})}$$

What is the problem

We may never see enough data for estimating these

Markov Assumption

Simplifying Assumption: Use only the previous word

$$P(\text{office} \mid \text{about fifteen minutes from}) \approx P(\text{office} \mid \text{from})$$

Markov Assumption

Simplifying Assumption: Use only the previous word

$$P(\text{office} \mid \text{about fifteen minutes from}) \approx P(\text{office} \mid \text{from})$$

Or the couple previous words

$$P(\text{office} \mid \text{about fifteen minutes from}) \approx P(\text{office} \mid \text{minutes from})$$

Markov Assumption

More Formally: kth order Markov Model

Chain Rule:

$$P(w_1 w_2 \dots w_n) = \prod_i P(w_i | w_1 w_2 \dots w_{i-1})$$

Markov Assumption

More Formally: kth order Markov Model

Chain Rule:

$$P(w_1 w_2 \dots w_n) = \prod_i P(w_i | w_1 w_2 \dots w_{i-1})$$

Using Markov Assumption: only k previous words

$$P(w_1 w_2 \dots w_n) \approx \prod_i P(w_i | w_{i-k} \dots w_{i-1})$$

Markov Assumption

More Formally: kth order Markov Model

Chain Rule:

$$P(w_1 w_2 \dots w_n) = \prod_i P(w_i | w_1 w_2 \dots w_{i-1})$$

Using Markov Assumption: only k previous words

$$P(w_1 w_2 \dots w_n) \approx \prod_i P(w_i | w_{i-k} \dots w_{i-1})$$

We approximate each component in the product

$$P(w_i | w_1 w_2 \dots w_{i-1}) \approx P(w_i | w_{i-k} \dots w_{i-1})$$

N-Gram Models

$P(\text{office} \mid \text{about fifteen minutes from})$

An N -gram model uses only $N - 1$ words of prior context.

N-Gram Models

$P(\text{office} \mid \text{about fifteen minutes from})$

An N -gram model uses only $N - 1$ words of prior context.

- Unigram: $P(\text{office})$
- Bigram: $P(\text{office} \mid \text{from})$
- Trigram: $P(\text{office} \mid \text{minutes from})$

N-Gram Models

$P(\text{office} \mid \text{about fifteen minutes from})$

An N -gram model uses only $N - 1$ words of prior context.

- Unigram: $P(\text{office})$
- Bigram: $P(\text{office} \mid \text{from})$
- Trigram: $P(\text{office} \mid \text{minutes from})$

Markov model and Language Model

N-Gram Models

$P(\text{office} \mid \text{about fifteen minutes from})$

An N -gram model uses only $N - 1$ words of prior context.

- Unigram: $P(\text{office})$
- Bigram: $P(\text{office} \mid \text{from})$
- Trigram: $P(\text{office} \mid \text{minutes from})$

Markov model and Language Model

An N -gram model is an $N - 1$ -order Markov Model

N-Gram Models

- We can extend to trigrams, 4-grams, 5-grams
- In general, an insufficient model of language:

N-Gram Models

- We can extend to trigrams, 4-grams, 5-grams
- In general, an insufficient model of language:
language has long-distance dependencies:
“The computer which I had just put into the machine room on the fifth floor **crashed**.”

N-Gram Models

- We can extend to trigrams, 4-grams, 5-grams
- In general, an insufficient model of language:
language has long-distance dependencies:
“The computer which I had just put into the machine room on the fifth floor **crashed.**”
- In most of the applications, we can get away with N-gram models

Estimating N-grams probabilities

NPTEL

Estimating N-grams probabilities

Maximum Likelihood Estimate

Value that makes the observed data the “most probable”

$$P(w_i|w_{i-1}) = \frac{\text{count}(w_{i-1}, w_i)}{\text{count}(w_{i-1})}$$

Estimating N-grams probabilities

Maximum Likelihood Estimate

Value that makes the observed data the “most probable”

$$P(w_i|w_{i-1}) = \frac{\text{count}(w_{i-1}, w_i)}{\text{count}(w_{i-1})}$$

$$P(w_i|w_{i-1}) = \frac{c(w_{i-1}, w_i)}{c(w_{i-1})}$$

An Example

$$P(w_i|w_{i-1}) = \frac{c(w_{i-1}, w_i)}{c(w_{i-1})}$$

<s>I am here </s>
<s>who am I </s>
<s>I would like to know </s>

An Example

$$P(w_i | w_{i-1}) = \frac{c(w_{i-1}, w_i)}{c(w_{i-1})}$$

<s>I am here </s>
<s>who am I </s>
<s>I would like to know </s>

Estimating bigrams

P(I|<s>) =

P(</s>|here) =

P(would | I) =

P(here | am) =

P(know | like) =

An Example

$$P(w_i | w_{i-1}) = \frac{c(w_{i-1}, w_i)}{c(w_{i-1})}$$

<s>I am here </s>
<s>who am I </s>
<s>I would like to know </s>

Estimating bigrams

$$P(I | <s>) = 2/3$$

$$P(</s> | \text{here}) = 1$$

$$P(\text{would} | I) = 1/3$$

$$P(\text{here} | \text{am}) = 1/2$$

$$P(\text{know} | \text{like}) = 0$$

Bigram counts from 9222 Restaurant Sentences

	i	want	to	eat	chinese	food	lunch	spend
i	5	827	0	9	0	0	0	2
want	2	0	608	1	6	6	5	1
to	2	0	4	686	2	0	6	211
eat	0	0	2	0	16	2	42	0
chinese	1	0	0	0	0	82	1	0
food	15	0	15	0	1	4	0	0
lunch	2	0	0	0	0	1	0	0
spend	1	0	1	0	0	0	0	0

Computing bigram probabilities

Normalize by unigrams

i	want	to	eat	chinese	food	lunch	spend
2533	927	2417	746	158	1093	341	278

Computing bigram probabilities

Normalize by unigrams

i	want	to	eat	chinese	food	lunch	spend
2533	927	2417	746	158	1093	341	278

Bigram Probabilities

	i	want	to	eat	chinese	food	lunch	spend
i	0.002	0.33	0	0.0036	0	0	0	0.00079
want	0.0022	0	0.66	0.0011	0.0065	0.0065	0.0054	0.0011
to	0.00083	0	0.0017	0.28	0.00083	0	0.0025	0.087
eat	0	0	0.0027	0	0.021	0.0027	0.056	0
chinese	0.0063	0	0	0	0	0.52	0.0063	0
food	0.014	0	0.014	0	0.00092	0.0037	0	0
lunch	0.0059	0	0	0	0	0.0029	0	0
spend	0.0036	0	0.0036	0	0	0	0	0

Computing Sentence Probabilities

$P(< s > I \text{ want english food } </ s >)$

$$= P(I | < s >) \times P(\text{want} | I) \times P(\text{english} | \text{want}) \times P(\text{food} | \text{english}) \times P(</ s > | \text{food})$$

Computing Sentence Probabilities

$P(< s > I \text{ want english food } </ s >)$

$$\begin{aligned} &= P(I | < s >) \times P(\text{want} | I) \times P(\text{english} | \text{want}) \times P(\text{food} | \text{english}) \times P(</ s > | \text{food}) \\ &= 0.000031 \end{aligned}$$

What knowledge does n-gram represent?

- $P(\text{english}|\text{want}) = .0011$
- $P(\text{chinese}|\text{want}) = .0065$
- $P(\text{to}|\text{want}) = .66$
- $P(\text{eat} | \text{to}) = .28$
- $P(\text{food} | \text{to}) = 0$
- $P(\text{want} | \text{spend}) = 0$
- $P(\text{i} | <\text{s}>) = .25$

Practical Issues

Everything in log space

- Avoids underflow

Practical Issues

Everything in log space

- Avoids underflow
- Adding is faster than multiplying

Practical Issues

Everything in log space

- Avoids underflow
- Adding is faster than multiplying

$$\log(p_1 \times p_2 \times p_3 \times p_4) = \log p_1 + \log p_2 + \log p_3 + \log p_4$$

Handling zeros

Use smoothing

Language Modeling Toolkit

SRILM

<http://www.speech.sri.com/projects/srilm/>

Google N-grams

Number of tokens: 1,024,908,267,229

Number of sentences: 95,119,665,584

Number of unigrams: 13,588,391

Number of bigrams: 314,843,401

Number of trigrams: 977,069,902

Number of fourgrams: 1,313,818,354

Number of fivegrams: 1,176,470,663

[http://googleresearch.blogspot.in/2006/08/
all-our-n-gram-are-belong-to-you.html](http://googleresearch.blogspot.in/2006/08/all-our-n-gram-are-belong-to-you.html)

Example from the 4-gram data

serve as the inspector 66

serve as the inspiration 1390

serve as the installation 136

serve as the institute 187

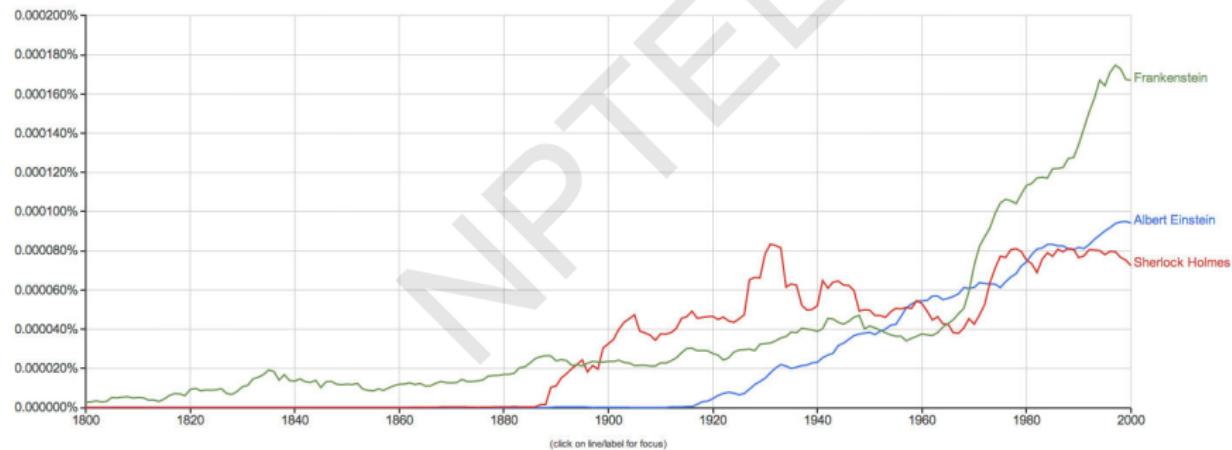
serve as the institution 279

serve as the institutional 461

Google books Ngram Data

Google books Ngram Viewer

Graph these comma-separated phrases: case-insensitive
between and from the corpus with smoothing of



Evaluation of Language Models, Basic Smoothing

Pawan Goyal

CSE, IITKGP

Week 2: Lecture 5

Evaluating Language Model

Does it prefer good sentences to bad sentences?

Assign higher probability to real (or frequently observed) sentences than ungrammatical (or rarely observed) ones

Evaluating Language Model

Does it prefer good sentences to bad sentences?

Assign higher probability to real (or frequently observed) sentences than ungrammatical (or rarely observed) ones

Training and Test Corpora

- Parameters of the model are trained on a large corpus of text, called **training set**.
- Performance is tested on a disjoint (held-out) **test data** using an **evaluation metric**

Extrinsic evaluation of N-grams models

Comparison of two models, A and B

- Use each model for one or more tasks: *spelling corrector, speech recognizer, machine translation*
- Get accuracy values for A and B
- Compare accuracy for A and B

Intrinsic evaluation: Perplexity

Intuition: The Shannon Game

How well can we predict the next word?

Intrinsic evaluation: Perplexity

Intuition: The Shannon Game

How well can we predict the next word?

- I always order pizza with cheese and ...
- The president of India is ...
- I wrote a ...

Intrinsic evaluation: Perplexity

Intuition: The Shannon Game

How well can we predict the next word?

- I always order pizza with cheese and ...
- The president of India is ...
- I wrote a ...

Unigram model doesn't work for this game.

Intrinsic evaluation: Perplexity

Intuition: The Shannon Game

How well can we predict the next word?

- I always order pizza with cheese and ...
- The president of India is ...
- I wrote a ...

Unigram model doesn't work for this game.

A better model of text

is one which assigns a higher probability to the actual word

Perplexity

The best language model is one that best predicts an unseen test set

Perplexity ($PP(W)$)

Perplexity is the inverse probability of the test data, normalized by the number of words:

Perplexity

The best language model is one that best predicts an unseen test set

Perplexity ($PP(W)$)

Perplexity is the inverse probability of the test data, normalized by the number of words:

$$PP(W) = P(w_1 w_2 \dots w_N)^{-\frac{1}{N}}$$

Perplexity

The best language model is one that best predicts an unseen test set

Perplexity ($PP(W)$)

Perplexity is the inverse probability of the test data, normalized by the number of words:

$$PP(W) = P(w_1 w_2 \dots w_N)^{-\frac{1}{N}}$$

Applying chain Rule

$$PP(W) = \left(\prod \frac{1}{P(w_i | w_1 \dots w_{i-1})} \right)^{\frac{1}{N}}$$

Perplexity

The best language model is one that best predicts an unseen test set

Perplexity ($PP(W)$)

Perplexity is the inverse probability of the test data, normalized by the number of words:

$$PP(W) = P(w_1 w_2 \dots w_N)^{-\frac{1}{N}}$$

Applying chain Rule

$$PP(W) = \left(\prod \frac{1}{P(w_i | w_1 \dots w_{i-1})} \right)^{\frac{1}{N}}$$

For bigrams

$$PP(W) = \left(\prod \frac{1}{P(w_i | w_{i-1})} \right)^{\frac{1}{N}}$$

Example: A Simple Scenario

- Consider a sentence consisting of N random digits

Example: A Simple Scenario

- Consider a sentence consisting of N random digits
- Find the perplexity of this sentence as per a model that assigns a probability $p = 1/10$ to each digit.

Example: A Simple Scenario

- Consider a sentence consisting of N random digits
- Find the perplexity of this sentence as per a model that assigns a probability $p = 1/10$ to each digit.

$$PP(W) = P(w_1 w_2 \dots w_N)^{-\frac{1}{N}}$$

Example: A Simple Scenario

- Consider a sentence consisting of N random digits
- Find the perplexity of this sentence as per a model that assigns a probability $p = 1/10$ to each digit.

$$\begin{aligned} PP(W) &= P(w_1 w_2 \dots w_N)^{-\frac{1}{N}} \\ &= \left(\left(\frac{1}{10} \right)^N \right)^{-\frac{1}{N}} \end{aligned}$$

Example: A Simple Scenario

- Consider a sentence consisting of N random digits
- Find the perplexity of this sentence as per a model that assigns a probability $p = 1/10$ to each digit.

$$\begin{aligned} PP(W) &= P(w_1 w_2 \dots w_N)^{-\frac{1}{N}} \\ &= \left(\left(\frac{1}{10} \right)^N \right)^{-\frac{1}{N}} \\ &= \left(\frac{1}{10} \right)^{-1} \\ &= 10 \end{aligned}$$

Lower perplexity = better model

WSJ Corpus

Training: 38 million words

Test: 1.5 million words

Lower perplexity = better model

WSJ Corpus

Training: 38 million words

Test: 1.5 million words

N-gram Order	Unigram	Bigram	Trigram
Perplexity	962	170	109

Lower perplexity = better model

WSJ Corpus

Training: 38 million words

Test: 1.5 million words

N-gram Order	Unigram	Bigram	Trigram
Perplexity	962	170	109

Unigram perplexity: 962?

The model is as confused on test data as if it had to choose uniformly and independently among 962 possibilities for each word.

The Shannon Visualization Method

Use the language model to generate word sequences

The Shannon Visualization Method

Use the language model to generate word sequences

- Choose a random bigram
 $(\langle s \rangle, w)$ as per its probability

The Shannon Visualization Method

Use the language model to generate word sequences

- Choose a random bigram
 $(\langle s \rangle, w)$ as per its probability
- Choose a random bigram
 (w, x) as per its probability

The Shannon Visualization Method

Use the language model to generate word sequences

- Choose a random bigram
 $(\langle s \rangle, w)$ as per its probability
- Choose a random bigram
 (w, x) as per its probability
- And so on until we choose
 $\langle /s \rangle$

The Shannon Visualization Method

Use the language model to generate word sequences

- Choose a random bigram ($< s >$, w) as per its probability
- Choose a random bigram (w, x) as per its probability
- And so on until we choose $</ s >$

$< s >$ I
I want
want to
to eat
eat Chinese
Chinese food
food $</ s >$
I want to eat Chinese food

Shakespeare as Corpus

- $N = 884,647$ tokens, $V = 29,066$
 - Shakespeare produced 300,000 bigram types out of $V^2 = 844$ million possible bigrams.

Approximating Shakespeare

Unigram

To him swallowed confess hear both. Which. Of save on trail for are ay device and rote life have
Every enter now severally so, let
Hill he late speaks; or! a more to leg less first you enter
Are where exeunt and sighs have rise excellency took of.. Sleep knave we. near; vile like

Bigram

What means, sir. I confess she? then all sorts, he is trim, captain.
Why dost stand forth thy canopy, forsooth; he is this palpable hit the King Henry. Live king. Follow.
What we, hath got so she that I rest and sent to scold and nature bankrupt, nor the first gentleman?

Trigram

Sweet prince, Falstaff shall die. Harry of Monmouth's grave.
This shall forbid it should be branded, if renown made it empty.
Indeed the duke; and had a very good friend.
Fly, and will rid me these news of price. Therefore the sadness of parting, as they say, 'tis done.

Quadrigram

King Henry.What! I will go seek the traitor Gloucester. Exeunt some of the watch. A great banquet serv'd in;
Will you not tell me who I am?
It cannot be but so.
Indeed the short and the long. Marry, 'tis a noble Lepidus.

Problems with simple MLE estimate: zeros

NPTEL

Problems with simple MLE estimate: zeros

Training set

- ... denied the allegations
- ... denied the reports
- ... denied the claims
- ... denied the request

Problems with simple MLE estimate: zeros

Training set

- ... denied the allegations
- ... denied the reports
- ... denied the claims
- ... denied the request

Test Data

- ... denied the offer
- ... denied the loan

Problems with simple MLE estimate: zeros

Training set

- ... denied the allegations
- ... denied the reports
- ... denied the claims
- ... denied the request

Test Data

- ... denied the offer
- ... denied the loan

Zero probability n-grams

- $P(\text{offer} \mid \text{denied the}) = 0$

Problems with simple MLE estimate: zeros

Training set

- ... denied the allegations
- ... denied the reports
- ... denied the claims
- ... denied the request

Test Data

- ... denied the offer
- ... denied the loan

Zero probability n-grams

- $P(\text{offer} \mid \text{denied the}) = 0$
- The test set will be assigned a probability 0

Problems with simple MLE estimate: zeros

Training set

- ... denied the allegations
- ... denied the reports
- ... denied the claims
- ... denied the request

Test Data

- ... denied the offer
- ... denied the loan

Zero probability n-grams

- $P(\text{offer} \mid \text{denied the}) = 0$
- The test set will be assigned a probability 0
- And the perplexity can't be computed

Language Modeling: Smoothing

NPTEL

Language Modeling: Smoothing

With sparse statistics

$P(w | \text{denied the})$

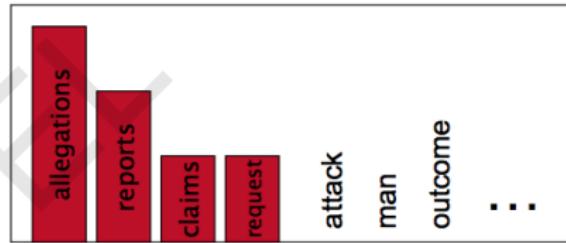
3 allegations

2 reports

1 claims

1 request

7 total



Language Modeling: Smoothing

With sparse statistics

$P(w | \text{denied the})$

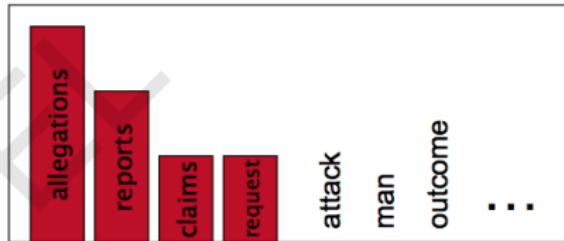
3 allegations

2 reports

1 claims

1 request

7 total



Steal probability mass to generalize better

$P(w | \text{denied the})$

2.5 allegations

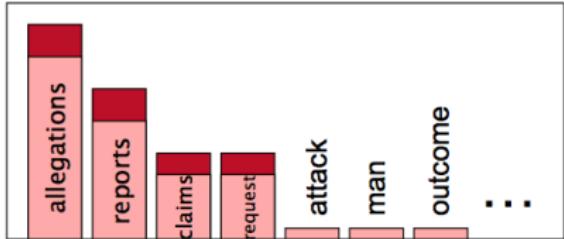
1.5 reports

0.5 claims

0.5 request

2 other

7 total



Laplace Smoothing (Add-one estimation)

- Pretend as if we saw each word (N-gram) one more time than we actually did

Laplace Smoothing (Add-one estimation)

- Pretend as if we saw each word (N-gram) one more time than we actually did
- Just add one to all the counts!

Laplace Smoothing (Add-one estimation)

- Pretend as if we saw each word (N-gram) one more time than we actually did
- Just add one to all the counts!
- MLE estimate for bigram: $P_{MLE}(w_i|w_{i-1}) = \frac{c(w_{i-1}, w_i)}{c(w_{i-1})}$

Laplace Smoothing (Add-one estimation)

- Pretend as if we saw each word (N-gram) one more time than we actually did
- Just add one to all the counts!
- MLE estimate for bigram: $P_{MLE}(w_i|w_{i-1}) = \frac{c(w_{i-1}, w_i)}{c(w_{i-1})}$
- Add-1 estimate: $P_{Add-1}(w_i|w_{i-1}) = \frac{c(w_{i-1}, w_i) + 1}{c(w_{i-1}) + V}$

Reconstituted counts as effect of smoothing

Effective bigram count ($c^*(w_{n-1}w_n)$)

$$\frac{c^*(w_{n-1}w_n)}{c(w_{n-1})} = \frac{c(w_{n-1}w_n) + 1}{c(w_{n-1}) + V}$$

Comparing with bigrams: Restaurant corpus

	i	want	to	eat	chinese	food	lunch	spend
i	5	827	0	9	0	0	0	2
want	2	0	608	1	6	6	5	1
to	2	0	4	686	2	0	6	211
eat	0	0	2	0	16	2	42	0
chinese	1	0	0	0	0	82	1	0
food	15	0	15	0	1	4	0	0
lunch	2	0	0	0	0	1	0	0
spend	1	0	1	0	0	0	0	0

Comparing with bigrams: Restaurant corpus

	i	want	to	eat	chinese	food	lunch	spend
i	5	827	0	9	0	0	0	2
want	2	0	608	1	6	6	5	1
to	2	0	4	686	2	0	6	211
eat	0	0	2	0	16	2	42	0
chinese	1	0	0	0	0	82	1	0
food	15	0	15	0	1	4	0	0
lunch	2	0	0	0	0	1	0	0
spend	1	0	1	0	0	0	0	0

	i	want	to	eat	chinese	food	lunch	spend
i	3.8	527	0.64	6.4	0.64	0.64	0.64	1.9
want	1.2	0.39	238	0.78	2.7	2.7	2.3	0.78
to	1.9	0.63	3.1	430	1.9	0.63	4.4	133
eat	0.34	0.34	1	0.34	5.8	1	15	0.34
chinese	0.2	0.098	0.098	0.098	0.098	8.2	0.2	0.098
food	6.9	0.43	6.9	0.43	0.86	2.2	0.43	0.43
lunch	0.57	0.19	0.19	0.19	0.19	0.38	0.19	0.19
spend	0.32	0.16	0.32	0.16	0.16	0.16	0.16	0.16

More general formulations: Add-k

$$P_{Add-k}(w_i | w_{i-1}) = \frac{c(w_{i-1}, w_i) + k}{c(w_{i-1}) + kV}$$

More general formulations: Add-k

$$P_{Add-k}(w_i|w_{i-1}) = \frac{c(w_{i-1}, w_i) + k}{c(w_{i-1}) + kV}$$

$$P_{Add-k}(w_i|w_{i-1}) = \frac{c(w_{i-1}, w_i) + m(\frac{1}{V})}{c(w_{i-1}) + m}$$

More general formulations: Add-k

$$P_{Add-k}(w_i|w_{i-1}) = \frac{c(w_{i-1}, w_i) + k}{c(w_{i-1}) + kV}$$

$$P_{Add-k}(w_i|w_{i-1}) = \frac{c(w_{i-1}, w_i) + m(\frac{1}{V})}{c(w_{i-1}) + m}$$

Unigram prior smoothing:

$$P_{UnigramPrior}(w_i|w_{i-1}) = \frac{c(w_{i-1}, w_i) + mP(w_i)}{c(w_{i-1}) + m}$$

More general formulations: Add-k

$$P_{Add-k}(w_i|w_{i-1}) = \frac{c(w_{i-1}, w_i) + k}{c(w_{i-1}) + kV}$$

$$P_{Add-k}(w_i|w_{i-1}) = \frac{c(w_{i-1}, w_i) + m(\frac{1}{V})}{c(w_{i-1}) + m}$$

Unigram prior smoothing:

$$P_{UnigramPrior}(w_i|w_{i-1}) = \frac{c(w_{i-1}, w_i) + mP(w_i)}{c(w_{i-1}) + m}$$

A good value of k or m ?

Can be optimized on held-out set

Language Modelling: Advanced Smoothing Models

Pawan Goyal

CSE, IITKGP

Week 3: Lecture 1

Advanced smoothing algorithms

Some Examples

- Good-Turing
- Kneser-Ney

Advanced smoothing algorithms

Some Examples

- Good-Turing
- Kneser-Ney

Good-Turing: Basic Intuition

Use the count of things we have seen once

- to help estimate the count of things we have never seen

N_c : Frequency of frequency c

Example Sentences

<s>I am here </s>

<s>who am I </s>

<s>I would like </s>

N_c : Frequency of frequency c

Example Sentences

<s>I am here </s>

<s>who am I </s>

<s>I would like </s>

Computing N_c

I	3	
am	2	
here	1	$N_1 = 4$
who	1	$N_2 = 1$
would	1	$N_3 = 1$
like	1	

Good Turing Estimation

Idea

- Reallocate the probability mass of n -grams that occur $r + 1$ times in the training data to the n -grams that occur r times
- In particular, reallocate the probability mass of n -grams that were seen once to the n -grams that were never seen

Adjusted count

For each count c , an adjusted count c^* is computed as:

$$c^* = \frac{(c+1)N_{c+1}}{N_c}$$

where N_c is the number of n -grams seen exactly c times

Good Turing Estimation

Good Turing Smoothing

$$P_{GT}^*(\text{things with frequency } c) = \frac{c^*}{N}$$

$$c^* = \frac{(c+1)N_{c+1}}{N_c}$$

Good Turing Estimation

Good Turing Smoothing

$$P_{GT}^*(\text{things with frequency } c) = \frac{c^*}{N}$$

$$c^* = \frac{(c+1)N_{c+1}}{N_c}$$

What if $c = 0$

$P_{GT}^*(\text{things with frequency } c) = \frac{N_1}{N}$ where N denotes the total number of bigrams that actually occur in training

Complications

What about words with high frequency?

- For small c , $N_c > N_{c+1}$
- For large c , too jumpy

Complications

What about words with high frequency?

- For small c , $N_c > N_{c+1}$
- For large c , too jumpy

Simple Good-Turing

Replace empirical N_k with a best-fit power law once counts get unreliable

Good-Turing numbers: Example

22 million words of AP Neswire

$$c^* = \frac{(c+1)N_{c+1}}{N_c}$$

Count c	Good Turing c*
0	.0000270
1	0.446
2	1.26
3	2.24
4	3.24
5	4.22
6	5.19
7	6.21
8	7.24
9	8.25

Good-Turing numbers: Example

22 million words of AP Neswire

$$c^* = \frac{(c+1)N_{c+1}}{N_c}$$

It looks like $c^* = c - 0.75$

Count c	Good Turing c*
0	.0000270
1	0.446
2	1.26
3	2.24
4	3.24
5	4.22
6	5.19
7	6.21
8	7.24
9	8.25

Absolute Discounting Interpolation

Why don't we just subtract 0.75 (or some d)?

Absolute Discounting Interpolation

Why don't we just subtract 0.75 (or some d)?

$$P_{AbsoluteDiscounting}(w_i | w_{i-1}) = \frac{c(w_{i-1}, w_i) - d}{c(w_{i-1})} + \lambda(w_{i-1})P(w_i)$$

Absolute Discounting Interpolation

Why don't we just subtract 0.75 (or some d)?

$$P_{AbsoluteDiscounting}(w_i|w_{i-1}) = \frac{c(w_{i-1}, w_i) - d}{c(w_{i-1})} + \lambda(w_{i-1})P(w_i)$$

We may keep some more values of d for counts 1 and 2

Absolute Discounting Interpolation

Why don't we just subtract 0.75 (or some d)?

$$P_{AbsoluteDiscounting}(w_i|w_{i-1}) = \frac{c(w_{i-1}, w_i) - d}{c(w_{i-1})} + \lambda(w_{i-1})P(w_i)$$

We may keep some more values of d for counts 1 and 2
But can we do better than using the regular unigram correct?

Kneser-Ney Smoothing

Intuition

- Shannon game: *I can't see without my reading ...: glasses/Francisco?*

Kneser-Ney Smoothing

Intuition

- Shannon game: *I can't see without my reading ...: glasses/Francisco?*
- “Francisco” more common than “glasses”

Kneser-Ney Smoothing

Intuition

- Shannon game: *I can't see without my reading ...: glasses/Francisco?*
- “Francisco” more common than “glasses”
- *But “Francisco” mostly follows “San”*

Kneser-Ney Smoothing

Intuition

- Shannon game: *I can't see without my reading ...: glasses/Francisco?*
- “Francisco” more common than “glasses”
- *But “Francisco” mostly follows “San”*

$P(w)$: “How likely is w ? ”

Instead, $P_{continuation}(w)$: “How likely is w to appear as a novel continuation?”

Kneser-Ney Smoothing

Intuition

- Shannon game: *I can't see without my reading ...*: glasses/Francisco?
- “Francisco” more common than “glasses”
- But “Francisco” mostly follows “San”

$P(w)$: “How likely is w ? ”

Instead, $P_{continuation}(w)$: “How likely is w to appear as a novel continuation?”

- For each word, count the number of bigram types it completes
- Every bigram type was a novel continuation the first time it was seen

Kneser-Ney Smoothing

Intuition

- Shannon game: *I can't see without my reading ...: glasses/Francisco?*
- “Francisco” more common than “glasses”
- *But “Francisco” mostly follows “San”*

$P(w)$: “How likely is w ? ”

Instead, $P_{continuation}(w)$: “How likely is w to appear as a novel continuation?”

- For each word, count the number of bigram types it completes
- Every bigram type was a novel continuation the first time it was seen

$$P_{continuation}(w) \propto |\{w_{i-1} : c(w_{i-1}, w) > 0\}|$$

Kneser-Ney Smoothing

How many times does w appear as a novel continuation?

$$P_{continuation}(w) \propto |\{w_{i-1} : c(w_{i-1}, w) > 0\}|$$

Kneser-Ney Smoothing

How many times does w appear as a novel continuation?

$$P_{continuation}(w) \propto |\{w_{i-1} : c(w_{i-1}, w) > 0\}|$$

Normalized by the total number of word bigram types

$$\frac{|\{(w_{j-1}, w_j) : c(w_{j-1}, w_j) > 0\}|}{|\{(w_{j-1}, w_j)\}|}$$

Kneser-Ney Smoothing

How many times does w appear as a novel continuation?

$$P_{continuation}(w) \propto |\{w_{i-1} : c(w_{i-1}, w) > 0\}|$$

Normalized by the total number of word bigram types

$$|\{(w_{j-1}, w_j) : c(w_{j-1}, w_j) > 0\}|$$

$$P_{continuation}(w) = \frac{|\{w_{i-1} : c(w_{i-1}, w) > 0\}|}{|\{(w_{j-1}, w_j) : c(w_{j-1}, w_j) > 0\}|}$$

Kneser-Ney Smoothing

How many times does w appear as a novel continuation?

$$P_{continuation}(w) \propto |\{w_{i-1} : c(w_{i-1}, w) > 0\}|$$

Normalized by the total number of word bigram types

$$|\{(w_{j-1}, w_j) : c(w_{j-1}, w_j) > 0\}|$$

$$P_{continuation}(w) = \frac{|\{w_{i-1} : c(w_{i-1}, w) > 0\}|}{|\{(w_{j-1}, w_j) : c(w_{j-1}, w_j) > 0\}|}$$

A frequent word (Francisco) occurring in only one context (San) will have a low continuation probability

Kneser-Ney Smoothing

$$P_{KN}(w_i|w_{i-1}) = \frac{\max(c(w_{i-1}, w_i) - d, 0)}{c(w_{i-1})} + \lambda(w_{i-1}) P_{continuation}(w_i)$$

Kneser-Ney Smoothing

$$P_{KN}(w_i|w_{i-1}) = \frac{\max(c(w_{i-1}, w_i) - d, 0)}{c(w_{i-1})} + \lambda(w_{i-1}) P_{continuation}(w_i)$$

λ is a normalizing constant

$$\lambda(w_{i-1}) = \frac{d}{c(w_{i-1})} |\{w : c(w_{i-1}, w) > 0\}|$$

Model Combination

As N increases

- The power (expressiveness) of an N-gram model increases

Model Combination

As N increases

- The power (expressiveness) of an N-gram model increases
- But the ability to estimate accurate parameters from sparse data decreases (i.e. the smoothing problem gets worse).

As N increases

- The power (expressiveness) of an N-gram model increases
- But the ability to estimate accurate parameters from sparse data decreases (i.e. the smoothing problem gets worse).

A general approach is to combine the results of multiple N-gram models.

Backoff and Interpolation

It might help to use less context

Backoff and Interpolation

It might help to use less context

when you haven't learned much about larger contexts

Backoff and Interpolation

It might help to use less context

when you haven't learned much about larger contexts

Backoff

- use trigram if you have good evidence
- otherwise bigram, otherwise unigram

Backoff and Interpolation

It might help to use less context

when you haven't learned much about larger contexts

Backoff

- use trigram if you have good evidence
- otherwise bigram, otherwise unigram

Interpolation

mix unigram, bigram, trigram

Backoff

Estimating $P(w_i|w_{i-2}w_{i-1})$

- If we do not have counts to compute $P(w_i|w_{i-2}w_{i-1})$ estimate this using the bigram probability $P(w_i|w_{i-1})$

Backoff

Estimating $P(w_i|w_{i-2}w_{i-1})$

- If we do not have counts to compute $P(w_i|w_{i-2}w_{i-1})$ estimate this using the bigram probability $P(w_i|w_{i-1})$
- If we do not have counts to compute $P(w_i|w_{i-1})$, estimate this using the unigram probability $P(w_i)$

Backoff

Estimating $P(w_i|w_{i-2}w_{i-1})$

- If we do not have counts to compute $P(w_i|w_{i-2}w_{i-1})$ estimate this using the bigram probability $P(w_i|w_{i-1})$
- If we do not have counts to compute $P(w_i|w_{i-1})$, estimate this using the unigram probability $P(w_i)$

$$P_{bo}(w_i|w_{i-2}w_{i-1}) =$$

- $\hat{P}(w_i|w_{i-2}w_{i-1})$, if $c(w_{i-2}w_{i-1}w_i) > 0$
- $\lambda(w_{i-1}w_{i-2})P_{bo}(w_i|w_{i-1})$, otherwise

$$\text{where } P_{bo}(w_i|w_{i-1}) =$$

- $\hat{P}(w_i|w_{i-1})$ if $c(w_{i-1}w_i) > 0$
- $\lambda(w_{n-1})\hat{P}(w_n)$, otherwise

Example Problem

In a corpus, suppose there are 4 words, a , b , c , and d . You are provided with the following counts.

n-gram	count	n-gram	count	n-gram	count
aba	4	ba	5	a	8
abb	0	bb	3	b	9
abc	0	bc	0	c	8
abd	0	bd	0	d	7

Use the recursive definition of backoff smoothing to obtain the probability distribution, $P_{backoff}(w_n | w_{n-2} w_{n-1})$, where $w_{n-1} = b$ and $w_{n-2} = a$. Also assume that $\hat{P}(x) = P(x) - 1/8$.

Linear Interpolation

Simple Interpolation

$$\tilde{P}(w_n | w_{n-1} w_{n-2}) = \lambda_1 P(w_n | w_{n-1} w_{n-2}) + \lambda_2 P(w_n | w_{n-1}) + \lambda_3 P(w_n)$$

Linear Interpolation

Simple Interpolation

$$\tilde{P}(w_n | w_{n-1} w_{n-2}) = \lambda_1 P(w_n | w_{n-1} w_{n-2}) + \lambda_2 P(w_n | w_{n-1}) + \lambda_3 P(w_n)$$

$$\sum_i \lambda_i = 1$$

Linear Interpolation

Simple Interpolation

$$\tilde{P}(w_n | w_{n-1} w_{n-2}) = \lambda_1 P(w_n | w_{n-1} w_{n-2}) + \lambda_2 P(w_n | w_{n-1}) + \lambda_3 P(w_n)$$

$$\sum_i \lambda_i = 1$$

Lambdas conditional on context

$$\begin{aligned}\tilde{P}(w_n | w_{n-1} w_{n-2}) &= \lambda_1(w_{n-2}, w_{n-1}) P(w_n | w_{n-1} w_{n-2}) \\ &\quad + \lambda_2(w_{n-2}, w_{n-1}) P(w_n | w_{n-1}) + \lambda_3(w_{n-2}, w_{n-1}) P(w_n)\end{aligned}$$

Setting the lambda values

Use a held-out corpus

Choose λ s to maximize the probability of held-out data:

- Find the N-gram probabilities on the training data
- Search for λ s that give the largest probability to held-out data

Computational Morphology

Pawan Goyal

CSE, IITKGP

Week 3: Lecture 2

Morphology

Morphology studies the internal structure of words, how words are built up from smaller meaningful units called **morphemes**

NPTEL

Morphology

Morphology studies the internal structure of words, how words are built up from smaller meaningful units called **morphemes**

dogs

- 2 morphemes, 'dog' and 's'
- 's' is a plural marker on nouns

Morphology

Morphology studies the internal structure of words, how words are built up from smaller meaningful units called **morphemes**

dogs

- 2 morphemes, 'dog' and 's'
- 's' is a plural marker on nouns

unladylike

3 morphemes

- un- 'not'
- lady 'well-behaved woman'
- -like 'having the characteristic of'

Allomorphs

Variants of the same morpheme, but cannot be replaced by one another

Example

- opposite: un-happy, in-comprehensible, im-possible, ir-rational

Bound and Free Morphemes

Bound

Cannot appear as a word by itself.

-s (*dog-s*), -ly (*quick-ly*), -ed (*walk-ed*)

Bound and Free Morphemes

Bound

Cannot appear as a word by itself.

-s (*dog-s*), -ly (*quick-ly*), -ed (*walk-ed*)

Free

Can appear as a word by itself; often can combine with other morphemes too.

house (*house-s*), *walk* (*walk-ed*), *of*, *the*, or

Stems and Affixes

Stems and Affixes

- Stems (roots): The core meaning bearing units
- Affixes: Bits and pieces adhering to stems to change their meanings and grammatical functions

Stems and Affixes

Stems and Affixes

- Stems (roots): The core meaning bearing units
- Affixes: Bits and pieces adhering to stems to change their meanings and grammatical functions

Mostly, stems are free morphemes and affixes are bound morphemes

Types of affixes

NPTEL

Types of affixes

- Prefix: un-, anti-, etc (a-, ati-, pra- etc.)
un-happy, pre-existing

Types of affixes

- Prefix: un-, anti-, etc (a-, ati-, pra- etc.)
un-happy, pre-existing
- Suffix: -ity, -ation, etc (-taa, -ke, -ka etc.)
talk-ing, quick-ly

Types of affixes

- Prefix: un-, anti-, etc (a-, ati-, pra- etc.)
un-happy, pre-existing
- Suffix: -ity, -ation, etc (-taa, -ke, -ka etc.)
talk-ing, quick-ly
- Infix: 'n' in '*vindati*' (he knows), as contrasted with *vid* (to know).
Philippines: basa 'read' → b-um-asa 'read'
English:

Types of affixes

- Prefix: un-, anti-, etc (a-, ati-, pra- etc.)
un-happy, pre-existing
- Suffix: -ity, -ation, etc (-taa, -ke, -ka etc.)
talk-ing, quick-ly
- Infix: 'n' in '*vindati*' (he knows), as contrasted with *vid* (to know).
Philippines: basa 'read' → b-um-asa 'read'
English: abso-bloody-lutely (emphasis)

Types of affixes

- Prefix: un-, anti-, etc (a-, ati-, pra- etc.)
un-happy, pre-existing
- Suffix: -ity, -ation, etc (-taa, -ke, -ka etc.)
talk-ing, quick-ly
- Infix: 'n' in '*vindati*' (he knows), as contrasted with *vid* (to know).
Philippines: basa 'read' → b-um-asa 'read'
English: abso-bloody-lutely (emphasis)
- Circumfixes - precedes and follows the stem
Dutch: berg 'mountain', ge-berg-te 'mountains'

Content and functional morphemes

Content morphemes

Carry some semantic content

car, -able, un-

Content and functional morphemes

Content morphemes

Carry some semantic content

car, -able, un-

Functional morphemes

Provide grammatical information

-s (plural), -s (3rd singular)

Inflectional and Derivational Morphology

Two different kind of relationship among words

NPTEL

Inflectional and Derivational Morphology

Two different kind of relationship among words

Inflectional morphology

Grammatical: number, tense, case, gender

Creates new forms of the same word: *bring, brought, brings, bringing*

Inflectional and Derivational Morphology

Two different kind of relationship among words

Inflectional morphology

Grammatical: number, tense, case, gender

Creates new forms of the same word: *bring, brought, brings, bringing*

Derivational morphology

Creates new words by changing part-of-speech: *logic, logical, illogical, illogicality, logician*

Inflectional and Derivational Morphology

Two different kind of relationship among words

Inflectional morphology

Grammatical: number, tense, case, gender

Creates new forms of the same word: *bring, brought, brings, bringing*

Derivational morphology

Creates new words by changing part-of-speech: *logic, logical, illogical, illogicality, logician*

Fairly systematic but some derivations missing: *sincere - sincerity, scarce - scarcity, curious - curiosity, fierce - fiercity?*

Morphological processes

Concatenation

Adding continuous affixes - the most common process:

- hope+less, un+happy, anti+capital+ist+s

Morphological processes

Concatenation

Adding continuous affixes - the most common process:

- hope+less, un+happy, anti+capital+ist+s

Often, there are phonological/graphemic changes on morpheme boundaries:

- book + s [s], shoe + s [z]
- happy +er → happier

Morphological processes

Reduplication: part of the word or the entire word is doubled

Morphological processes

Reduplication: part of the word or the entire word is doubled

- Nama: 'go' (look), 'go-go' (examine with attention)

Morphological processes

Reduplication: part of the word or the entire word is doubled

- Nama: 'go' (look), 'go-go' (examine with attention)
- Tagalog: 'basa' (read), 'ba-basa'(will read)

Morphological processes

Reduplication: part of the word or the entire word is doubled

- Nama: 'go' (look), 'go-go' (examine with attention)
- Tagalog: 'basa' (read), 'ba-basa' (will read)
- Sanskrit: 'pac' (cook), 'papāca' (perfect form, cooked)

Morphological processes

Reduplication: part of the word or the entire word is doubled

- Nama: 'go' (look), 'go-go' (examine with attention)
- Tagalog: 'basa' (read), 'ba-basa' (will read)
- Sanskrit: 'pac' (cook), 'papāca' (perfect form, cooked)
- Phrasal reduplication (Telugu): *pillavādu naḍustū naḍustū paḍi pōyādu*
(The child fell down while walking)

Morphological processes

Suppletion

'irregular' relation between the words
go - went, good - better

Morphological processes

Suppletion

'irregular' relation between the words
go - went, good - better

Morpheme internal changes

The word changes internally
sing - sang - sung, man - men, goose - geese

Word Formation

Compounding

Words formed by combining two or more words

Example in English:

- Adj + Adj → Adj: bitter-sweet
- N + N → N: rain-bow
- V + N → V: pick-pocket
- P + V → V: over-do

Word Formation

Compounding

Words formed by combining two or more words

Example in English:

- Adj + Adj → Adj: bitter-sweet
- N + N → N: rain-bow
- V + N → V: pick-pocket
- P + V → V: over-do

Particular to languages

room-temperature: Hindi translation?

Word Formation

Compounding

Words formed by combining two or more words

Example in English:

- Adj + Adj → Adj: bitter-sweet
- N + N → N: rain-bow
- V + N → V: pick-pocket
- P + V → V: over-do

Particular to languages

room-temperature: Hindi translation?

Word Formation

Acronyms

laser: Light Amplification by Simulated Emission of Radiation

Word Formation

Acronyms

laser: Light Amplification by Simulated Emission of Radiation

Blending

Parts of two different words are combined

- breakfast + lunch → brunch
- smoke + fog → smog
- motor + hotel → motel

Word Formation

Acronyms

laser: Light Amplification by Simulated Emission of Radiation

Blending

Parts of two different words are combined

- breakfast + lunch → brunch
- smoke + fog → smog
- motor + hotel → motel

Clipping

Longer words are shortened

Word Formation

Acronyms

laser: Light Amplification by Simulated Emission of Radiation

Blending

Parts of two different words are combined

- breakfast + lunch → brunch
- smoke + fog → smog
- motor + hotel → motel

Clipping

Longer words are shortened

doctor, laboratory, advertisement, dormitory, examination, bicycle, refrigerator

Processing morphology

- Lemmatization: word → lemma
 $\text{saw} \rightarrow \{\text{see}, \text{saw}\}$

Processing morphology

- Lemmatization: word → lemma
 $\text{saw} \rightarrow \{\text{see}, \text{saw}\}$
- Morphological analysis : word → setOf(lemma +tag)
 $\text{saw} \rightarrow \{ \langle \text{see}, \text{verb.past} \rangle, \langle \text{saw}, \text{noun.sg} \rangle \}$

Processing morphology

- Lemmatization: word → lemma
 $\text{saw} \rightarrow \{\text{see}, \text{saw}\}$
- Morphological analysis : word → setOf(lemma +tag)
 $\text{saw} \rightarrow \{ \langle \text{see}, \text{verb.past} \rangle, \langle \text{saw}, \text{noun.sg} \rangle \}$
- Tagging: word → tag, considers context
 $\text{Peter saw her} \rightarrow \{ \langle \text{see}, \text{verb.past} \rangle \}$

Processing morphology

- Lemmatization: word → lemma
 $\text{saw} \rightarrow \{\text{see}, \text{saw}\}$
- Morphological analysis : word → setOf(lemma +tag)
 $\text{saw} \rightarrow \{ \langle \text{see}, \text{verb.past} \rangle, \langle \text{saw}, \text{noun.sg} \rangle \}$
- Tagging: word → tag, considers context
 $\text{Peter saw her} \rightarrow \{ \langle \text{see}, \text{verb.past} \rangle \}$
- Morpheme segmentation: de-nation-al-iz-ation

Processing morphology

- Lemmatization: word → lemma
 $\text{saw} \rightarrow \{\text{see}, \text{saw}\}$
- Morphological analysis : word → setOf(lemma +tag)
 $\text{saw} \rightarrow \{ \langle \text{see}, \text{verb.past} \rangle, \langle \text{saw}, \text{noun.sg} \rangle \}$
- Tagging: word → tag, considers context
 $\text{Peter saw her} \rightarrow \{ \langle \text{see}, \text{verb.past} \rangle \}$
- Morpheme segmentation: de-nation-al-iz-ation
- Generation: see + verb.past → saw

What are the applications?

- Text-to-speech synthesis:
lead:

What are the applications?

- Text-to-speech synthesis:
lead: verb or noun?
read:

What are the applications?

- Text-to-speech synthesis:
lead: verb or noun?
read: present or past?
 - Search and information retrieval
 - Machine translation, grammar correction

Morphological Analysis

Input	Morphological Parsed Output
cats	cat +N +PL
cat	cat +N +SG
cities	city +N +PL
geese	goose +N +PL
goose	(goose +N +SG) or (goose +V)
gooses	goose +V +3SG
merging	merge +V +PRES-PART
caught	(catch +V +PAST-PART) or (catch +V +PAST)

Morphological Analysis

Input	Morphological Parsed Output
cats	cat +N +PL
cat	cat +N +SG
cities	city +N +PL
geese	goose +N +PL
goose	(goose +N +SG) or (goose +V)
gooses	goose +V +3SG
merging	merge +V +PRES-PART
caught	(catch +V +PAST-PART) or (catch +V +PAST)

Goal

To take input forms like those in the first column and produce output forms like those in the second column.

Output contains stem and additional information; +N for noun, +SG for singular, +PL for plural, +V for verb etc.

Issues involved

boy → boys

Issues involved

boy → boys

fly → flies → flies (y→ i rule)

Issues involved

boy → boys

fly → flys → flies (y→ i rule)

Toiling → toil

Issues involved

boy → boys

fly → flys → flies (y→ i rule)

Toiling → toil

Duckling → duckl?

Issues involved

boy → boys

fly → flys → flies (y→ i rule)

Toiling → toil

Duckling → duckl?

- Getter → get + er
- Doer → do + er

Issues involved

boy → boys

fly → flys → flies (y→ i rule)

Toiling → toil

Duckling → duckl?

- Getter → get + er
- Doer → do + er
- Beer → be + er?

Knowledge Required

Knowledge of stems or roots

Duck is a possible root, not duckl.

We need a dictionary (lexicon)

Knowledge Required

Knowledge of stems or roots

Duck is a possible root, not *duck!*.

We need a dictionary (lexicon)

Morphotactics

Which class of morphemes follow other classes of morphemes inside the word?

Ex: plural morpheme follows the noun

Knowledge Required

Knowledge of stems or roots

Duck is a possible root, not *duck!*.

We need a dictionary (lexicon)

Morphotactics

Which class of morphemes follow other classes of morphemes inside the word?

Ex: plural morpheme follows the noun

Only some endings go on some words

- *Do+er*: ok
- *Be+er*: not so

Knowledge Required

Knowledge of stems or roots

Duck is a possible root, not *duckl*.

We need a dictionary (lexicon)

Morphotactics

Which class of morphemes follow other classes of morphemes inside the word?

Ex: plural morpheme follows the noun

Only some endings go on some words

- *Do+er*: ok
- *Be+er*: not so

Spelling change rules

Adjust the surface form using spelling change rules

- *Get + er → getter*

Why can't this be put in a big lexicon?

- English: just 317,477 forms from 90,196 lexical entries, a ratio of 3.5:1

Why can't this be put in a big lexicon?

- English: just 317,477 forms from 90,196 lexical entries, a ratio of 3.5:1
- Sanskrit: 11 million forms from a lexicon of 170,000 entries, a ratio of 64.7:1

Why can't this be put in a big lexicon?

- English: just 317,477 forms from 90,196 lexical entries, a ratio of 3.5:1
- Sanskrit: 11 million forms from a lexicon of 170,000 entries, a ratio of 64.7:1
- New forms can be created, compounding etc.

One of the most common methods is finite-state-machines

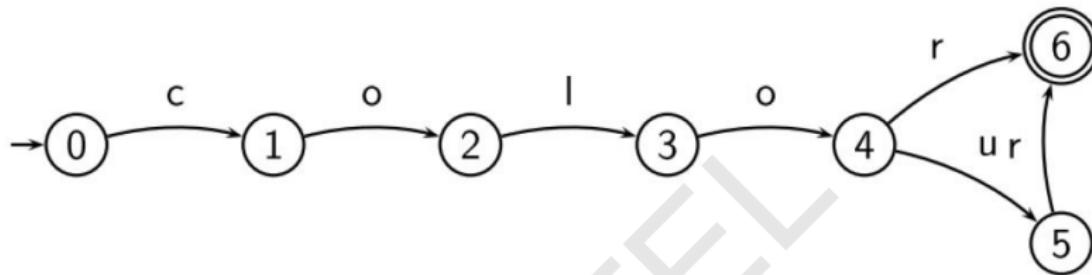
Finite-state methods for morphology

Pawan Goyal

CSE, IITKGP

Week 3: Lecture 3

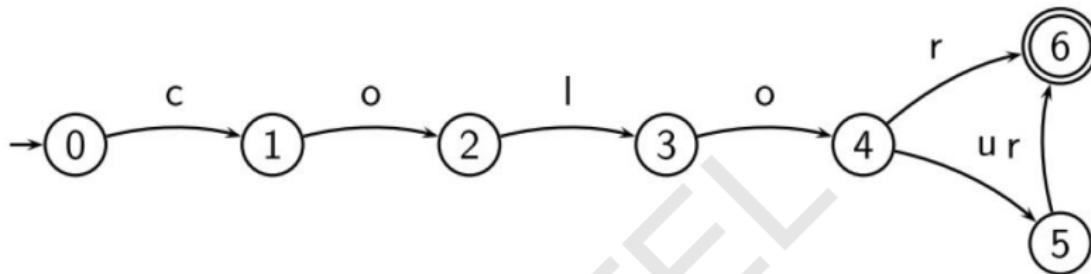
Finite State Automaton (FSA)



What is FSA?

- A kind of directed graph

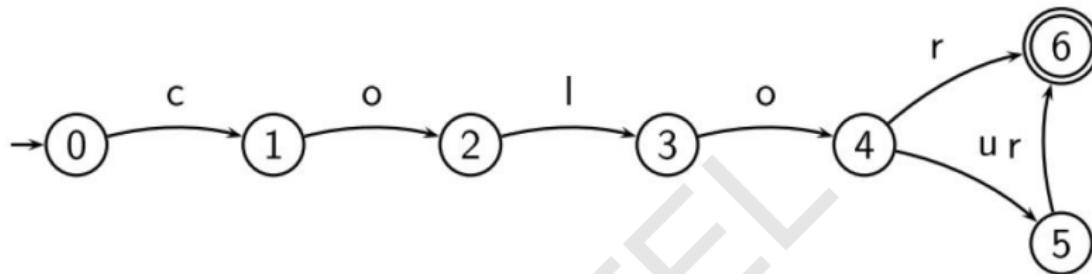
Finite State Automaton (FSA)



What is FSA?

- A kind of directed graph
- Nodes are called states, edges are labeled with symbols (possibly empty ϵ)

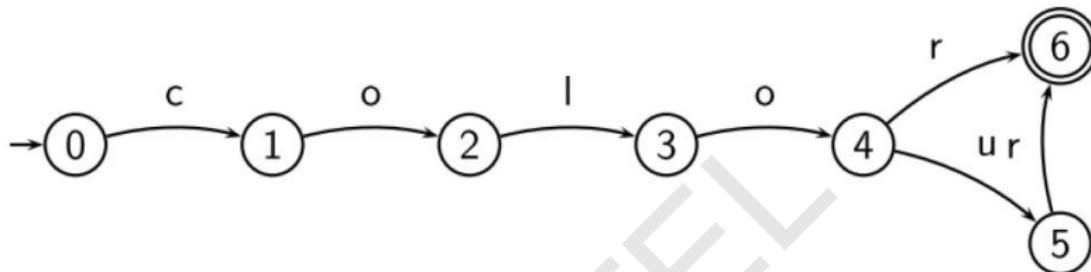
Finite State Automaton (FSA)



What is FSA?

- A kind of directed graph
- Nodes are called states, edges are labeled with symbols (possibly empty ϵ)
- Start state and accepting states

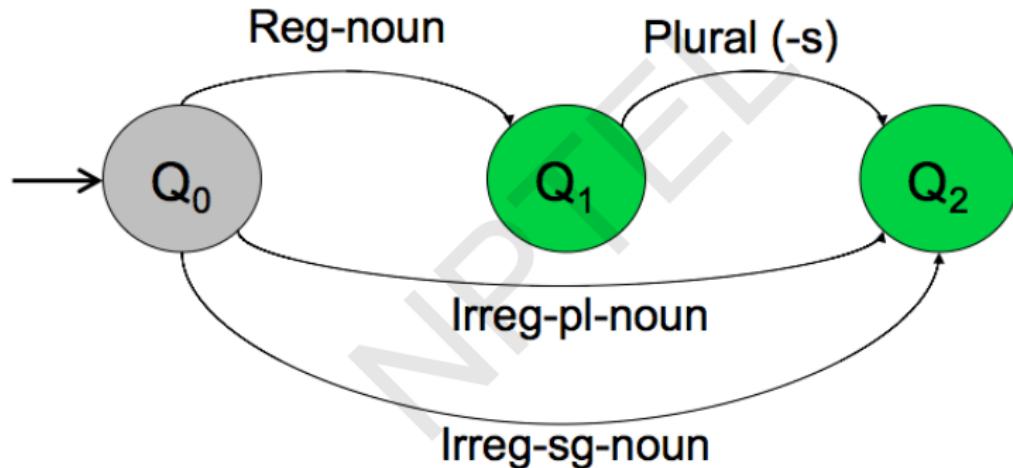
Finite State Automaton (FSA)



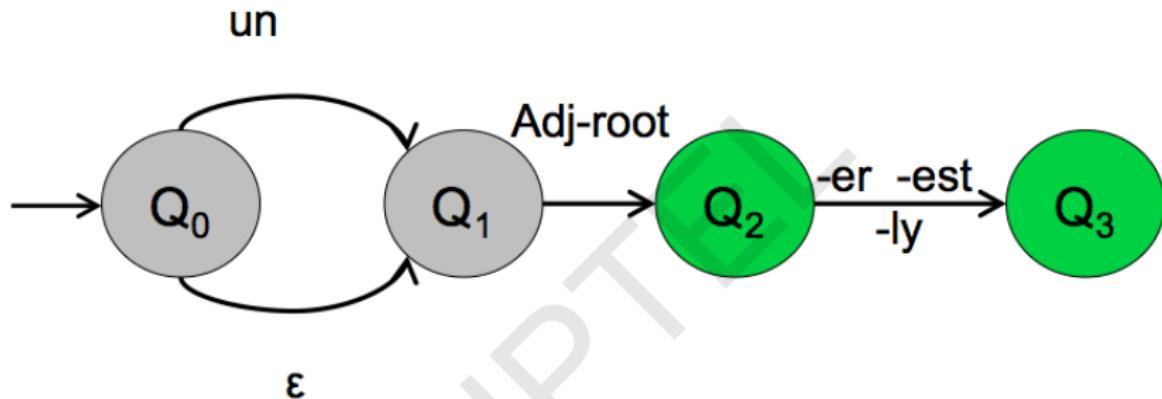
What is FSA?

- A kind of directed graph
- Nodes are called states, edges are labeled with symbols (possibly empty ϵ)
- Start state and accepting states
- Recognizes regular languages, i.e., languages specified by regular expressions

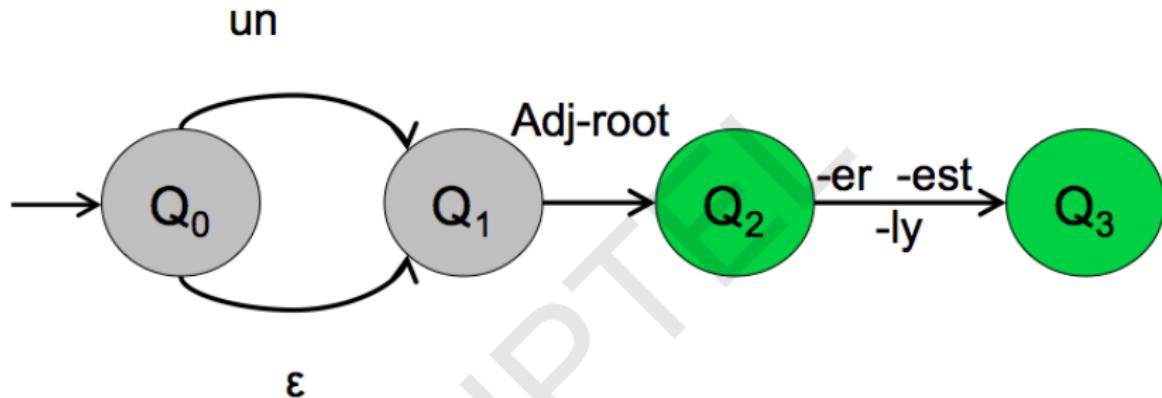
FSA for nominal inflection in English



FSA for English Adjectives



FSA for English Adjectives



Word modeled

happy, happier, happiest, real, unreal, cool, coolly, clear, clearly, unclear,
unclearly, ...

Morphotactics

- The last two examples model some parts of the English morphotactics
- But what about the information about regular and irregular roots?

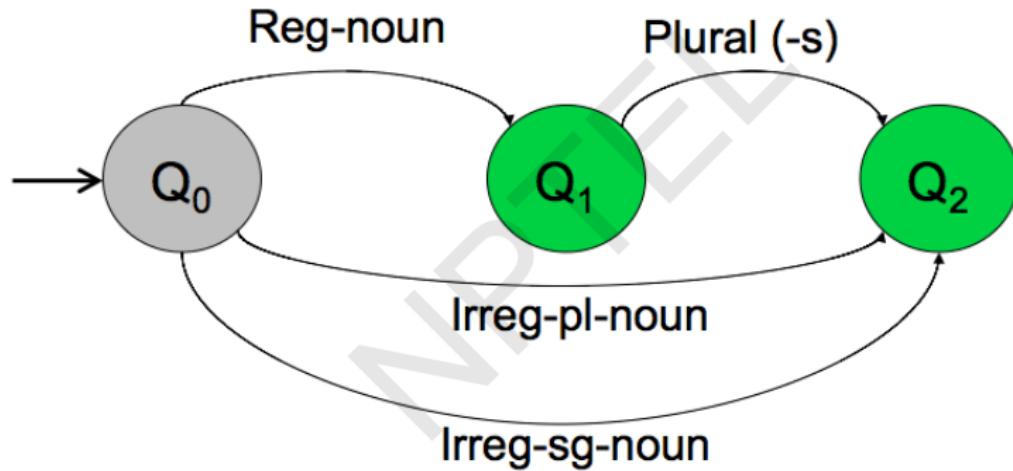
Morphotactics

- The last two examples model some parts of the English morphotactics
- But what about the information about regular and irregular roots?

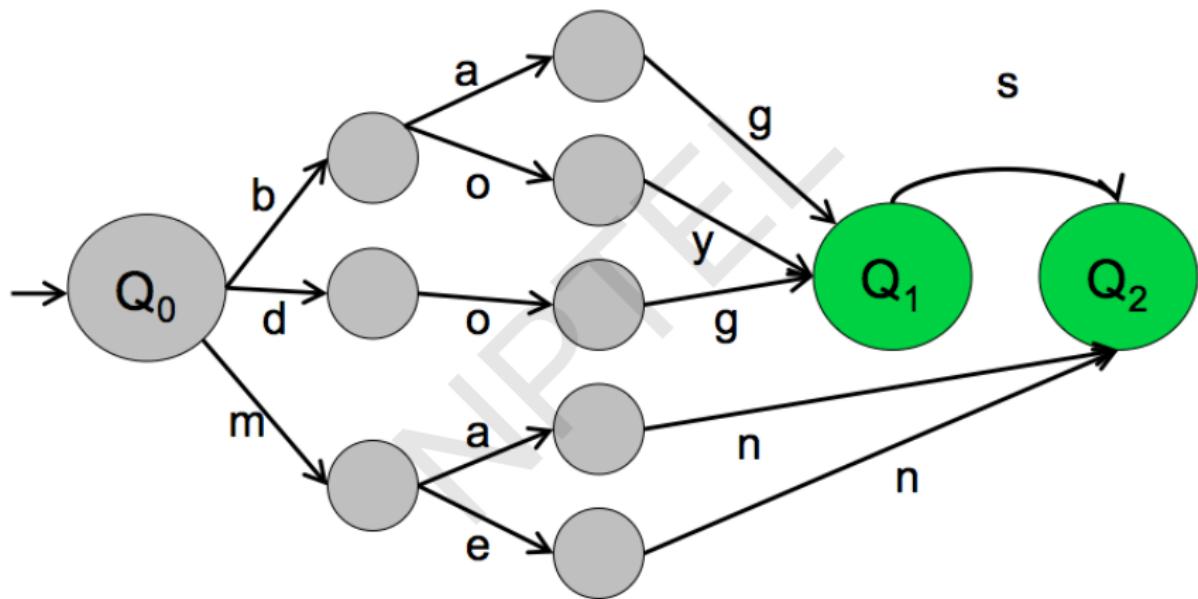
Lexicon

Can we include the lexicon in the FSA?

FSA for nominal inflection in English



After adding a mini-lexicon



Some properties of FSAs: Elegance

- Recognizing problem can be solved in linear time (independent of the size of the automaton)
- There is an algorithm to transform each automaton into a unique equivalent automaton with the least number of states
- An FSA is deterministic iff it has no empty (ϵ) transition and for each state and each symbol, there is at most one applicable transition
- Every non-deterministic automaton can be transformed into a deterministic one

But ...

FSAs are language recognizers/generators.

But ...

FSAs are language recognizers/generators.

We need transducers to build Morphological Analyzers

But ...

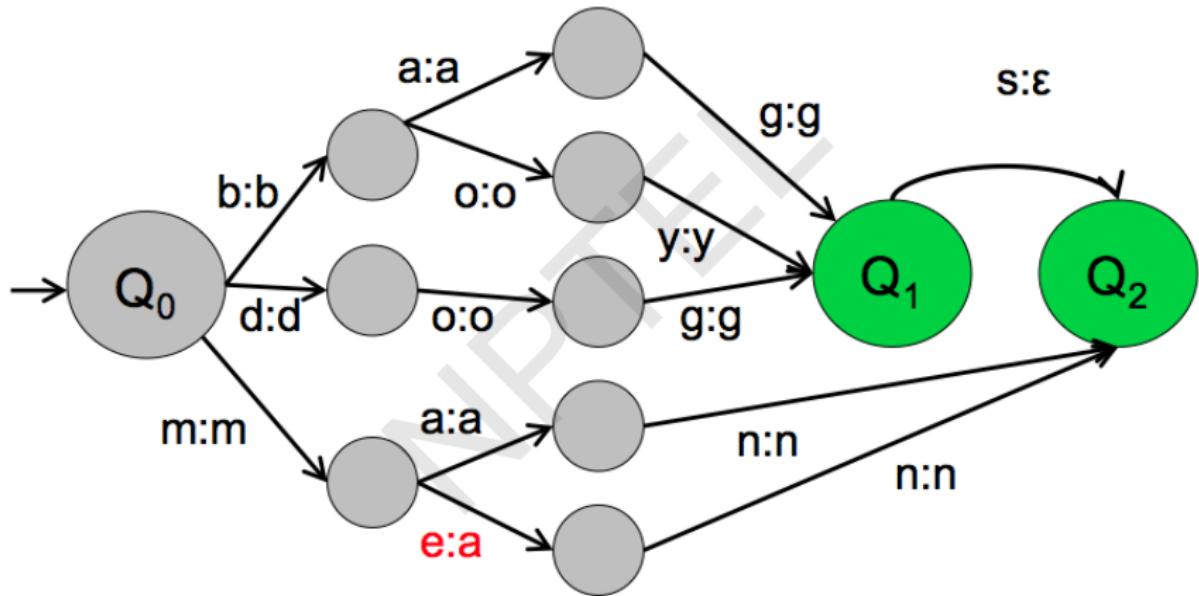
FSA are language recognizers/generators.

We need transducers to build Morphological Analyzers

Finite State Transducers

- Translate strings from one language to strings in another language
- Like FSA, but each edge is associated with two strings

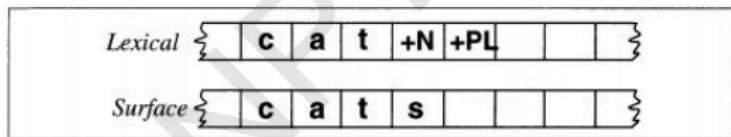
An example FST



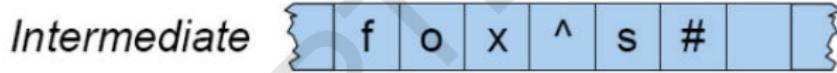
Two-level morphology

Given the input *cats*, we would like to output *cat+N+PL*, telling us that cat is a plural noun.

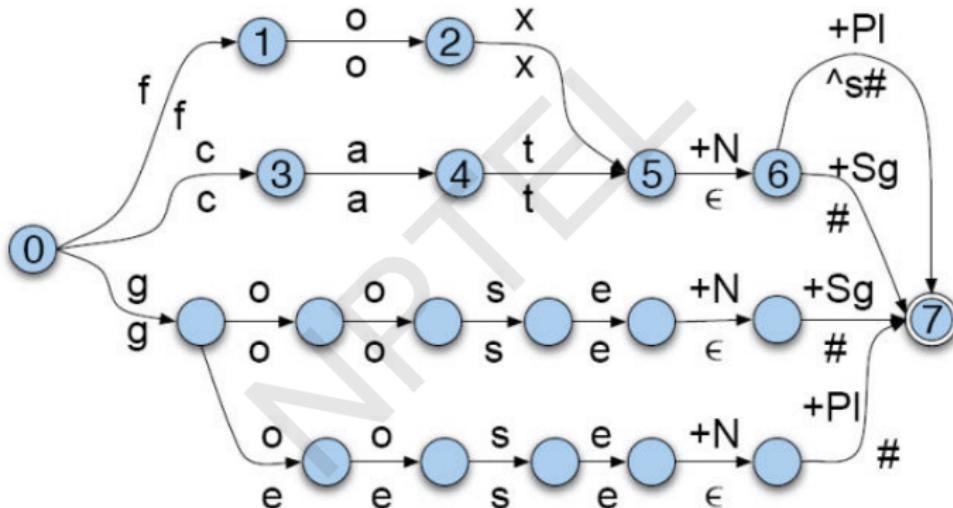
We do this via a version of **two-level morphology**, a correspondence between a lexical level (morphemes and features) to a surface level (actual spelling).



Intermediate tape for Spelling change rules

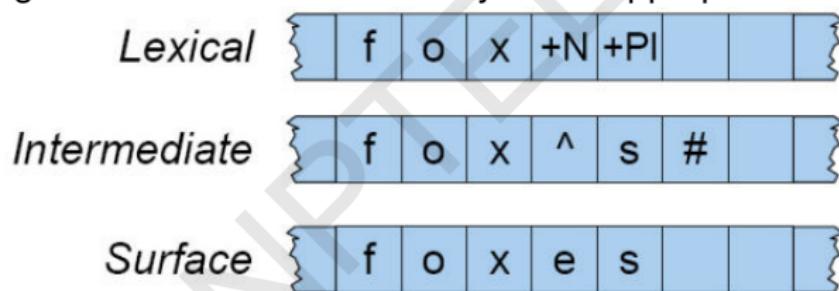


English Nominal Inflection FST



Spelling Handling

A spelling change rule would insert an e only in the appropriate environment.



Rule Handling

Rule Notation

$a \rightarrow b/c_d$: “rewrite a as b when it occurs between c and d .”

Morphological Analysis: Approaches

Two different ways to address phonological/graphemic variations

- Linguistic approach: A phonological component accompanying the simple concatenative process of attaching an ending
- Engineering approach: Phonological changes and irregularities are factored into endings and a higher number of paradigms

Different Approaches: Example from Czech

	woman	owl	draft	iceberg	vapor	fly
S1	žen-a	sov-a	skic-a	kr-a	pár-a	mouch-a
S2	žen-y	sov-y	skic-i	kr-y	pář-y	mouch-y
S3	žen-ě	sov-ě	skic-e	kř-e	pář-e	mouš-e
:						
P2	žen-0	sov-0	skic-0	ker-0	par-0	much-0

A linguistic approach

$$\begin{array}{llllll} \text{žen} + \begin{cases} \text{a} \\ \text{y} \\ \check{\text{e}} \\ 0 \end{cases} & \text{sov} + \begin{cases} \text{a} \\ \text{y} \\ \check{\text{e}} \\ 0 \end{cases} & \text{skic} + \begin{cases} \text{a} \\ \text{y} \\ \check{\text{e}} \\ 0 \end{cases} & \text{kr} + \begin{cases} \text{a} \\ \text{y} \\ \check{\text{e}} \\ 0 \end{cases} & \text{pár} + \begin{cases} \text{a} \\ \text{y} \\ \check{\text{e}} \\ 0 \end{cases} & \text{mouch} + \begin{cases} \text{a} \\ \text{y} \\ \check{\text{e}} \\ 0 \end{cases} \end{array}$$

An engineering approach

$$\begin{array}{llllll} \text{žen} + \begin{cases} \text{a} \\ \text{y} \\ \check{\text{e}} \\ 0 \end{cases} & \text{sov} + \begin{cases} \text{a} \\ \text{y} \\ \check{\text{e}} \\ 0 \end{cases} & \text{skic} + \begin{cases} \text{a} \\ \text{i} \\ \text{e} \\ 0 \end{cases} & \text{k} + \begin{cases} \text{ra} \\ \text{ry} \\ \check{\text{e}}\text{r} \\ \text{er} \end{cases} & \text{p} + \begin{cases} \text{ára} \\ \text{áry} \\ \check{\text{e}}\text{r} \\ \text{ar} \end{cases} & \text{m} + \begin{cases} \text{oucha} \\ \text{ouchy} \\ \text{ouše} \\ \text{uch} \end{cases} \end{array}$$

Tools Available

- AT&T FSM Library and Lextools

<http://www2.research.att.com/~fsmtools/fsm/>

- OpenFST (Google and NYU)

<http://www.openfst.org/>

Introduction to POS Tagging

Pawan Goyal

CSE, IITKGP

Week 3: Lecture 4

Part-of-Speech (POS) tagging

NPTEL

Part-of-Speech (POS) tagging

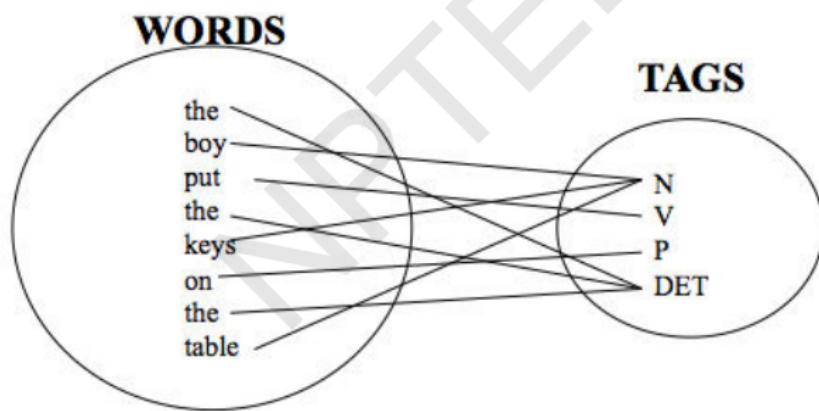
Task

Given a text of English, identify the parts of speech of each word

Part-of-Speech (POS) tagging

Task

Given a text of English, identify the parts of speech of each word



Parts of Speech: How many?

Open class words (content words)

- nouns, verbs, adjectives, adverbs
- mostly content-bearing: they refer to objects, actions, and features in the world
- *open class*, since new words are added all the time

Parts of Speech: How many?

Open class words (content words)

- nouns, verbs, adjectives, adverbs
- mostly content-bearing: they refer to objects, actions, and features in the world
- *open class*, since new words are added all the time

Closed class words

- pronouns, determiners, prepositions, connectives, ...
- there is a limited number of these
- *mostly functional*: to tie the concepts of a sentence together

POS examples

- N noun chair, bandwidth, pacing
- V verb study, debate, munch
- ADJ adj purple, tall, ridiculous
- ADV adverb unfortunately, slowly,
- P preposition of, by, to
- PRO pronoun I, me, mine
- DET determiner the, a, that, those

POS tagging: Choosing a tagset

- To do POS tagging, a standard set needs to be chosen

POS tagging: Choosing a tagset

- To do POS tagging, a standard set needs to be chosen
- Could pick very coarse tagsets
N, V, Adj, Adv

POS tagging: Choosing a tagset

- To do POS tagging, a standard set needs to be chosen
- Could pick very coarse tagsets
 N, V, Adj, Adv
- More commonly used set is finer grained, “UPenn TreeBank tagset”, 45 tags

POS tagging: Choosing a tagset

- To do POS tagging, a standard set needs to be chosen
- Could pick very coarse tagsets
 N , V , Adj , Adv
- More commonly used set is finer grained, “UPenn TreeBank tagset”, 45 tags

A Nice Tutorial on POS tags

<https://sites.google.com/site/partofspeechhelp/>

UPenn TreeBank POS tag set

Tag	Description	Example	Tag	Description	Example
CC	Coordin. Conjunction	<i>and, but, or</i>	SYM	Symbol	+%, &
CD	Cardinal number	<i>one, two, three</i>	TO	"to"	<i>to</i>
DT	Determiner	<i>a, the</i>	UH	Interjection	<i>ah, oops</i>
EX	Existential 'there'	<i>there</i>	VB	Verb, base form	<i>eat</i>
FW	Foreign word	<i>mea culpa</i>	VBD	Verb, past tense	<i>ate</i>
IN	Preposition/sub-conj	<i>of, in, by</i>	VBG	Verb, gerund	<i>eating</i>
JJ	Adjective	<i>yellow</i>	VBN	Verb, past participle	<i>eaten</i>
JJR	Adj., comparative	<i>bigger</i>	VBP	Verb, non-3sg pres	<i>eat</i>
JJS	Adj., superlative	<i>wildest</i>	VBZ	Verb, 3sg pres	<i>eats</i>
LS	List item marker	<i>1, 2, One</i>	WDT	Wh-determiner	<i>which, that</i>
MD	Modal	<i>can, should</i>	WP	Wh-pronoun	<i>what, who</i>
NN	Noun, sing. or mass	<i>llama</i>	WP\$	Possessive wh-	<i>whose</i>
NNS	Noun, plural	<i>llamas</i>	WRB	Wh-adverb	<i>how, where</i>
NNP	Proper noun, singular	<i>IBM</i>	\$	Dollar sign	\$
NNPS	Proper noun, plural	<i>Carolinas</i>	#	Pound sign	#
PDT	Predeterminer	<i>all, both</i>	"	Left quote	(“ or “)
POS	Possessive ending	<i>'s</i>	"	Right quote	(‘ or ’)
PRP	Personal pronoun	<i>I, you, he</i>	(Left parenthesis	([, (, {, <)
PRP\$	Possessive pronoun	<i>your, one's</i>)	Right parenthesis	(],), }, >)
RB	Adverb	<i>quickly, never</i>	,	Comma	,
RBR	Adverb, comparative	<i>faster</i>	.	Sentence-final punc	(. ! ?)
RBS	Adverb, superlative	<i>fastest</i>	:	Mid-sentence punc	(: ; ... – -)
RP	Particle	<i>up, off</i>			

Using the UPenn tagset

Example Sentence

The grand jury commented on a number of other topics.

Using the UPenn tagset

Example Sentence

The grand jury commented on a number of other topics.

POS tagged sentence

The/DT grand/JJ jury/NN commmented/VBD on/IN a/DT number/NN of/IN other/JJ topics/NNS ./.

Why is POS tagging hard?

NPTEL

Why is POS tagging hard?

Words often have more than one POS: back

- The back door:

Why is POS tagging hard?

Words often have more than one POS: back

- The back door: *back/JJ*
- On my back:

Why is POS tagging hard?

Words often have more than one POS: back

- The back door: *back/JJ*
- On my back: *back/NN*
- Win the voters back:

Why is POS tagging hard?

Words often have more than one POS: back

- The back door: *back/JJ*
- On my back: *back/NN*
- Win the voters back: *back/RB*
- Promised to back the bill:

Why is POS tagging hard?

Words often have more than one POS: back

- The back door: *back/JJ*
- On my back: *back/NN*
- Win the voters back: *back/RB*
- Promised to back the bill: *back/VB*

Why is POS tagging hard?

Words often have more than one POS: back

- The back door: *back/JJ*
- On my back: *back/NN*
- Win the voters back: *back/RB*
- Promised to back the bill: *back/VB*

POS tagging problem

To determine the POS tag for a particular instance of a word

Ambiguous word types in the Brown Corpus

Ambiguity in the Brown corpus

- 40% of word tokens are ambiguous
- 12% of word types are ambiguous

Ambiguous word types in the Brown Corpus

Ambiguity in the Brown corpus

- 40% of word tokens are ambiguous
- 12% of word types are ambiguous
- Breakdown of ambiguous word types:

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")

How bad is the ambiguity problem?

- One tag is usually more likely than the others.

How bad is the ambiguity problem?

- One tag is usually more likely than the others.

In the Brown corpus, *race* is a noun 98% of the time, and a verb 2% of the time

How bad is the ambiguity problem?

- One tag is usually more likely than the others.
In the Brown corpus, *race* is a noun 98% of the time, and a verb 2% of the time
- A tagger for English that simply chooses the most likely tag for each word can achieve good performance

How bad is the ambiguity problem?

- One tag is usually more likely than the others.
In the Brown corpus, *race* is a noun 98% of the time, and a verb 2% of the time
- A tagger for English that simply chooses the most likely tag for each word can achieve good performance
- Any new approach should be compared against the unigram baseline (assigning each token to its most likely tag)

Deciding the correct POS

Can be difficult even for people

- Mrs./NNP Shaefer/NNP never/RB got/VBD around/_ to/TO joining/VBG.
- All/DT we/PRP gotta/VBN do/VB is/VBZ go/VB around/_ the/DT corner/NN.
- Chateau/NNP Petrus/NNP costs/VBZ around/_ 2500/CD.

Deciding the correct POS

Can be difficult even for people

- Mrs./NNP Shaefer/NNP never/RB got/VBD around/RP to/TO joining/VBG.
- All/DT we/PRP gotta/VBN do/VB is/VBZ go/VB around/IN the/DT corner/NN.
- Chateau/NNP Petrus/NNP costs/VBZ around/RB 2500/CD.

Relevant knowledge for POS tagging

The word itself

- Some words may only be nouns, e.g. *arrow*
- Some words are ambiguous, e.g. *like, flies*
- Probabilities may help, if one tag is more likely than another

Relevant knowledge for POS tagging

The word itself

- Some words may only be nouns, e.g. *arrow*
- Some words are ambiguous, e.g. *like, flies*
- Probabilities may help, if one tag is more likely than another

Local context

- Two determiners rarely follow each other
- Two base form verbs rarely follow each other
- Determiner is almost always followed by adjective or noun

POS tagging: Two approaches

Rule-based Approach

- Assign each word in the input a list of potential POS tags
- Then winnow down this list to a single tag using hand-written rules

POS tagging: Two approaches

Rule-based Approach

- Assign each word in the input a list of potential POS tags
- Then winnow down this list to a single tag using hand-written rules

Statistical tagging

- Get a training corpus of tagged text, learn the transformation rules from the most frequent tags (TBL tagger)
- Probabilistic: Find the most likely sequence of tags T for a sequence of words W

TBL Tagger

Label the training set with most frequent tags

- The can was rusted.

TBL Tagger

Label the training set with most frequent tags

- The can was rusted.
- The/DT can/MD was/VBD rusted/VBD.

TBL Tagger

Label the training set with most frequent tags

- The can was rusted.
- The/DT can/MD was/VBD rusted/VBD.

Add transformation rules to reduce training mistakes

- MD →NN: DT_
- VBD→VBN: VBD_

Probabilistic Tagging: Two different families of models

Problem at hand

We have some data $\{(d, c)\}$ of paired observations d and hidden classes c .

Probabilistic Tagging: Two different families of models

Problem at hand

We have some data $\{(d, c)\}$ of paired observations d and hidden classes c .

Different instances of d and c

- **Part-of-Speech Tagging:**

Probabilistic Tagging: Two different families of models

Problem at hand

We have some data $\{(d, c)\}$ of paired observations d and hidden classes c .

Different instances of d and c

- **Part-of-Speech Tagging:** words are observed and tags are hidden.
- **Text Classification:**

Probabilistic Tagging: Two different families of models

Problem at hand

We have some data $\{(d, c)\}$ of paired observations d and hidden classes c .

Different instances of d and c

- **Part-of-Speech Tagging:** words are observed and tags are hidden.
- **Text Classification:** sentences/documents are observed and the category is hidden.

Probabilistic Tagging: Two different families of models

Problem at hand

We have some data $\{(d, c)\}$ of paired observations d and hidden classes c .

Different instances of d and c

- **Part-of-Speech Tagging:** words are observed and tags are hidden.
- **Text Classification:** sentences/documents are observed and the category is hidden.
Categories can be positive/negative for sentiments ..
sports/politics/business for documents ...

Probabilistic Tagging: Two different families of models

Problem at hand

We have some data $\{(d, c)\}$ of paired observations d and hidden classes c .

Different instances of d and c

- **Part-of-Speech Tagging:** words are observed and tags are hidden.
- **Text Classification:** sentences/documents are observed and the category is hidden.
Categories can be positive/negative for sentiments ..
sports/politics/business for documents ...

What gives rise to the two families?

Whether they generate the observed data from hidden stuff or the hidden structure given the data?

Generative vs. Conditional Models

Generative (Joint) Models

Generate the observed data from hidden stuff, i.e. put a probability over the observations given the class: $P(d, c)$ in terms of $P(d|c)$

Generative vs. Conditional Models

Generative (Joint) Models

Generate the observed data from hidden stuff, i.e. put a probability over the observations given the class: $P(d, c)$ in terms of $P(d|c)$
e.g. Naïve Bayes' classifiers, Hidden Markov Models etc.

Generative vs. Conditional Models

Generative (Joint) Models

Generate the observed data from hidden stuff, i.e. put a probability over the observations given the class: $P(d, c)$ in terms of $P(d|c)$
e.g. Naïve Bayes' classifiers, Hidden Markov Models etc.

Discriminative (Conditional) Models

Take the data as given, and put a probability over hidden structure given the data: $P(c|d)$

Generative vs. Conditional Models

Generative (Joint) Models

Generate the observed data from hidden stuff, i.e. put a probability over the observations given the class: $P(d, c)$ in terms of $P(d|c)$
e.g. Naïve Bayes' classifiers, Hidden Markov Models etc.

Discriminative (Conditional) Models

Take the data as given, and put a probability over hidden structure given the data: $P(c|d)$
e.g. Logistic regression, maximum entropy models, conditional random fields

Generative vs. Conditional Models

Generative (Joint) Models

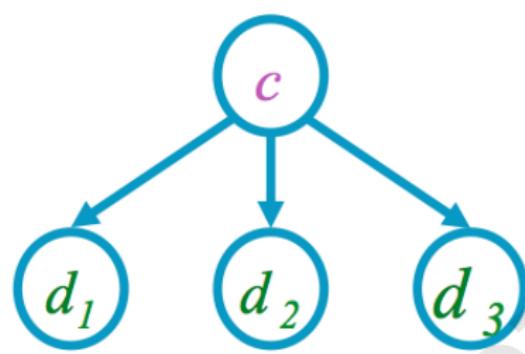
Generate the observed data from hidden stuff, i.e. put a probability over the observations given the class: $P(d, c)$ in terms of $P(d|c)$
e.g. Naïve Bayes' classifiers, Hidden Markov Models etc.

Discriminative (Conditional) Models

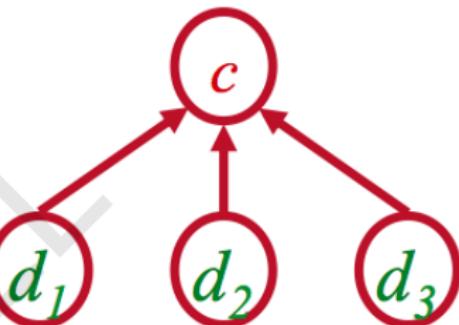
Take the data as given, and put a probability over hidden structure given the data: $P(c|d)$
e.g. Logistic regression, maximum entropy models, conditional random fields

SVMs, perceptron, etc. are discriminative classifiers but not directly probabilistic

Generative vs. Discriminative Models

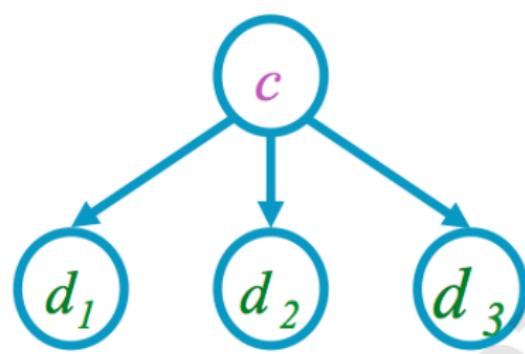


Naive Bayes

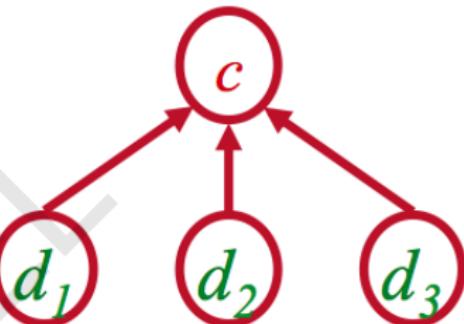


Logistic Regression

Generative vs. Discriminative Models



Naive Bayes



Logistic Regression

Joint vs. conditional likelihood

- A *joint* model gives probabilities $P(d, c)$ and tries to maximize this joint likelihood.
- A *conditional* model gives probabilities $P(c|d)$, taking the data as given and modeling only the conditional probability of the class.

Hidden Markov Models for POS Tagging

Pawan Goyal

CSE, IITKGP

Week 3: Lecture 5

Probabilistic Tagging

- $W = w_1 \dots w_n$ - words in the corpus (observed)
- $T = t_1 \dots t_n$ - the corresponding tags (unknown)

Probabilistic Tagging

- $W = w_1 \dots w_n$ - words in the corpus (observed)
- $T = t_1 \dots t_n$ - the corresponding tags (unknown)

Tagging: Probabilistic View (Generative Model)

Find

$$\hat{T} = \operatorname{argmax}_T P(T|W)$$

Probabilistic Tagging

- $W = w_1 \dots w_n$ - words in the corpus (observed)
- $T = t_1 \dots t_n$ - the corresponding tags (unknown)

Tagging: Probabilistic View (Generative Model)

Find

$$\begin{aligned}\hat{T} &= \operatorname{argmax}_T P(T|W) \\ &= \operatorname{argmax}_T \frac{P(W|T)P(T)}{P(W)}\end{aligned}$$

Probabilistic Tagging

- $W = w_1 \dots w_n$ - words in the corpus (observed)
- $T = t_1 \dots t_n$ - the corresponding tags (unknown)

Tagging: Probabilistic View (Generative Model)

Find

$$\begin{aligned}\hat{T} &= \operatorname{argmax}_T P(T|W) \\ &= \operatorname{argmax}_T \frac{P(W|T)P(T)}{P(W)} \\ &= \operatorname{argmax}_T P(W|T)P(T)\end{aligned}$$

Probabilistic Tagging

- $W = w_1 \dots w_n$ - words in the corpus (observed)
- $T = t_1 \dots t_n$ - the corresponding tags (unknown)

Tagging: Probabilistic View (Generative Model)

Find

$$\begin{aligned}\hat{T} &= \operatorname{argmax}_T P(T|W) \\ &= \operatorname{argmax}_T \frac{P(W|T)P(T)}{P(W)} \\ &= \operatorname{argmax}_T P(W|T)P(T) \\ &= \operatorname{argmax}_T \prod_i P(w_i|w_1 \dots w_{i-1}, t_1 \dots t_i)P(t_i|t_1 \dots t_{i-1})\end{aligned}$$

Further simplifications

$$\hat{T} = \operatorname{argmax}_T \prod_i P(w_i | w_1 \dots w_{i-1}, t_1 \dots t_i) P(t_i | t_1 \dots t_{i-1})$$

- The probability of a word appearing depends only on its own POS tag

Further simplifications

$$\hat{T} = \operatorname{argmax}_T \prod_i P(w_i | w_1 \dots w_{i-1}, t_1 \dots t_i) P(t_i | t_1 \dots t_{i-1})$$

- The probability of a word appearing depends only on its own POS tag
 $P(w_i | w_1 \dots w_{i-1}, t_1 \dots t_i) \approx P(w_i | t_i)$
- Bigram assumption: the probability of a tag appearing depends only on the previous tag

Further simplifications

$$\hat{T} = \operatorname{argmax}_T \prod_i P(w_i | w_1 \dots w_{i-1}, t_1 \dots t_i) P(t_i | t_1 \dots t_{i-1})$$

- The probability of a word appearing depends only on its own POS tag
 $P(w_i | w_1 \dots w_{i-1}, t_1 \dots t_i) \approx P(w_i | t_i)$
- Bigram assumption: the probability of a tag appearing depends only on the previous tag
 $P(t_i | t_1 \dots t_{i-1}) \approx P(t_i | t_{i-1})$

Further simplifications

$$\hat{T} = \operatorname{argmax}_T \prod_i P(w_i | w_1 \dots w_{i-1}, t_1 \dots t_i) P(t_i | t_1 \dots t_{i-1})$$

- The probability of a word appearing depends only on its own POS tag
 $P(w_i | w_1 \dots w_{i-1}, t_1 \dots t_i) \approx P(w_i | t_i)$
- Bigram assumption: the probability of a tag appearing depends only on the previous tag
 $P(t_i | t_1 \dots t_{i-1}) \approx P(t_i | t_{i-1})$
- Using these simplifications:

$$\hat{T} = \operatorname{argmax}_T \prod_i P(w_i | t_i) P(t_i | t_{i-1})$$

Computing the probability values

Tag Transition probabilities $p(t_i|t_{i-1})$

$$P(t_i|t_{i-1}) = \frac{C(t_{i-1}, t_i)}{C(t_{i-1})}$$

$$P(NN|DT) = \frac{C(DT, NN)}{C(DT)} = \frac{56,509}{116,454} = 0.49$$

Computing the probability values

Tag Transition probabilities $p(t_i|t_{i-1})$

$$P(t_i|t_{i-1}) = \frac{C(t_{i-1}, t_i)}{C(t_{i-1})}$$

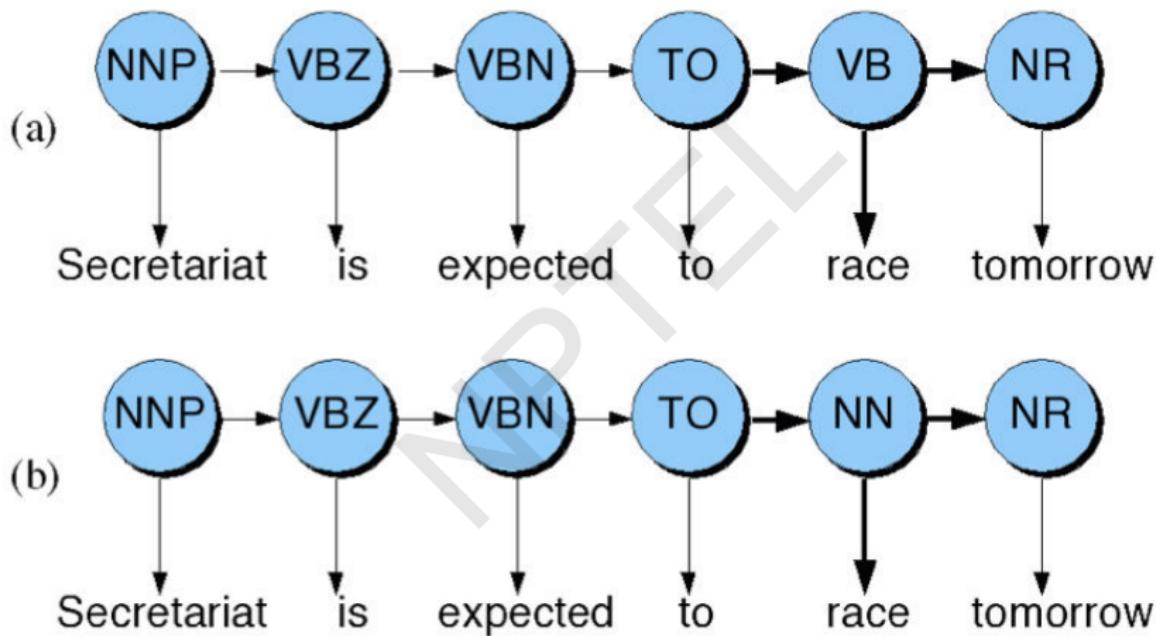
$$P(NN|DT) = \frac{C(DT, NN)}{C(DT)} = \frac{56,509}{116,454} = 0.49$$

Word Likelihood probabilities $p(w_i|t_i)$

$$P(w_i|t_i) = \frac{C(t_i, w_i)}{C(t_i)}$$

$$P(is|VBZ) = \frac{C(VBZ, is)}{C(VBZ)} = \frac{10,073}{21,627} = 0.47$$

Disambiguating “race”



Disambiguating “race”

Difference in probability due to

- $P(VB|TO)$ vs. $P(NN|TO)$
- $P(race|VB)$ vs. $P(race|NN)$
- $P(NR|VB)$ vs. $P(NR|NN)$

Disambiguating “race”

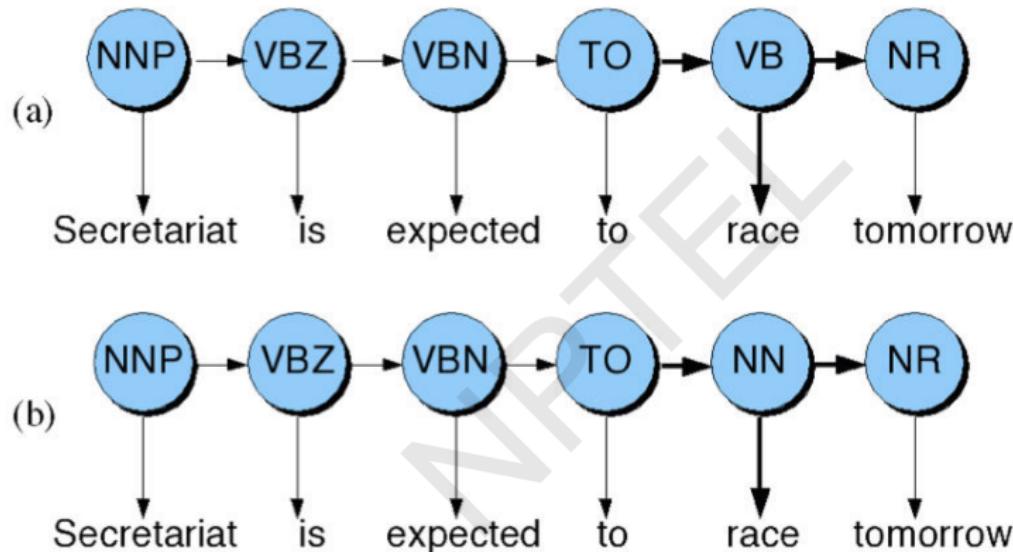
Difference in probability due to

- $P(VB|TO)$ vs. $P(NN|TO)$
- $P(race|VB)$ vs. $P(race|NN)$
- $P(NR|VB)$ vs. $P(NR|NN)$

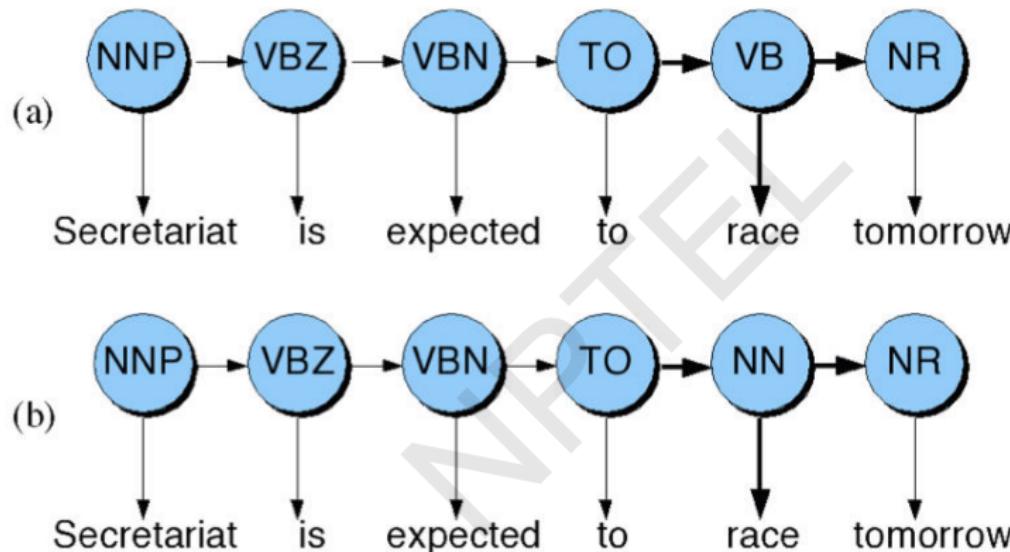
After computing the probabilities

- $P(NN|TO)P(NR|NN)P(race|NN) = 0.0047 \times 0.0012 \times 0.00057 = 0.00000000032$
- $P(VB|TO)P(NR|VB)P(race|VB) = 0.83 \times 0.0027 \times 0.00012 = 0.00000027$

What is this model?



What is this model?



This is a Hidden Markov Model

Hidden Markov Models

- Tag Transition probabilities $p(t_i|t_{i-1})$
- Word Likelihood probabilities (emissions) $p(w_i|t_i)$

Hidden Markov Models

- Tag Transition probabilities $p(t_i|t_{i-1})$
- Word Likelihood probabilities (emissions) $p(w_i|t_i)$
- What we have described with these probabilities is a hidden markov model.
- Let us quickly introduce the Markov Chain, or observable Markov Model.

Markov Chain = First-order Markov Model

Weather example

- Three types of weather: *sunny, rainy, foggy*

Markov Chain = First-order Markov Model

Weather example

- Three types of weather: *sunny, rainy, foggy*
- q_n : variable denoting the weather on the n^{th} day

Markov Chain = First-order Markov Model

Weather example

- Three types of weather: *sunny, rainy, foggy*
- q_n : variable denoting the weather on the n^{th} day
- We want to find the following conditional probabilities:

$$P(q_n|q_{n-1}, q_{n-2}, \dots, q_1)$$

Markov Chain = First-order Markov Model

Weather example

- Three types of weather: *sunny, rainy, foggy*
- q_n : variable denoting the weather on the n^{th} day
- We want to find the following conditional probabilities:

$$P(q_n|q_{n-1}, q_{n-2}, \dots, q_1)$$

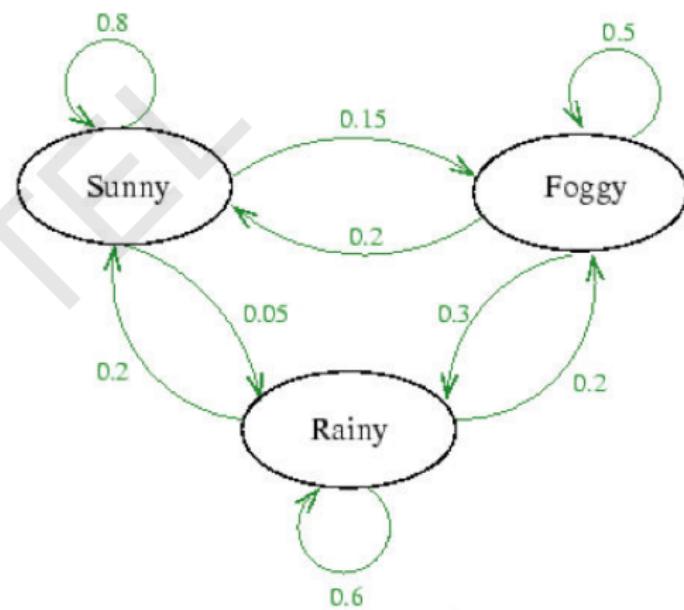
First-order Markov Assumption

$$P(q_n|q_{n-1}, q_{n-2}, \dots, q_1) = P(q_n|q_{n-1})$$

Markov Chain Transition Table

Table 1: Probabilities $p(q_{n+1}|q_n)$ of tomorrow's weather based on today's weather

		Tomorrow's weather		
		Sunny	Rainy	Foggy
Today's weather				
Sunny		0.8	0.05	0.15
Rainy		0.2	0.6	0.2
Foggy		0.2	0.3	0.5



Using Markov Chain

- Given that today the weather is sunny, what is the probability that tomorrow is sunny and day after is rainy?

Using Markov Chain

- Given that today the weather is sunny, what is the probability that tomorrow is sunny and day after is rainy?

$$P(q_2 = \text{sunny}, q_3 = \text{rainy} | q_1 = \text{sunny})$$

Using Markov Chain

- Given that today the weather is sunny, what is the probability that tomorrow is sunny and day after is rainy?

$$P(q_2 = \text{sunny}, q_3 = \text{rainy} | q_1 = \text{sunny})$$

$$= P(q_3 = \text{rainy} | q_2 = \text{sunny}, q_1 = \text{sunny}) \times P(q_2 = \text{sunny} | q_1 = \text{sunny})$$

Using Markov Chain

- Given that today the weather is sunny, what is the probability that tomorrow is sunny and day after is rainy?

$$P(q_2 = \text{sunny}, q_3 = \text{rainy} | q_1 = \text{sunny})$$

$$\begin{aligned}&= P(q_3 = \text{rainy} | q_2 = \text{sunny}, q_1 = \text{sunny}) \times P(q_2 = \text{sunny} | q_1 = \text{sunny}) \\&= P(q_3 = \text{rainy} | q_2 = \text{sunny}) \times P(q_2 = \text{sunny} | q_1 = \text{sunny}) \\&= 0.05 \times 0.8 \\&= 0.04\end{aligned}$$

Hidden Markov Model

- For Markov chains, the output symbols are the same as the states
'sunny' weather is both observable and state

Hidden Markov Model

- For Markov chains, the output symbols are the same as the states
'sunny' weather is both observable and state
- But in POS tagging

Hidden Markov Model

- For Markov chains, the output symbols are the same as the states
'sunny' weather is both observable and state
- But in POS tagging
The output symbols are words

Hidden Markov Model

- For Markov chains, the output symbols are the same as the states
'sunny' weather is both observable and state
- But in POS tagging
The output symbols are words
But the hidden states are POS tags
- A Hidden Markov Model is an extension of a Markov chain in which the output symbols are not the same as the states
- We don't know which state we are in

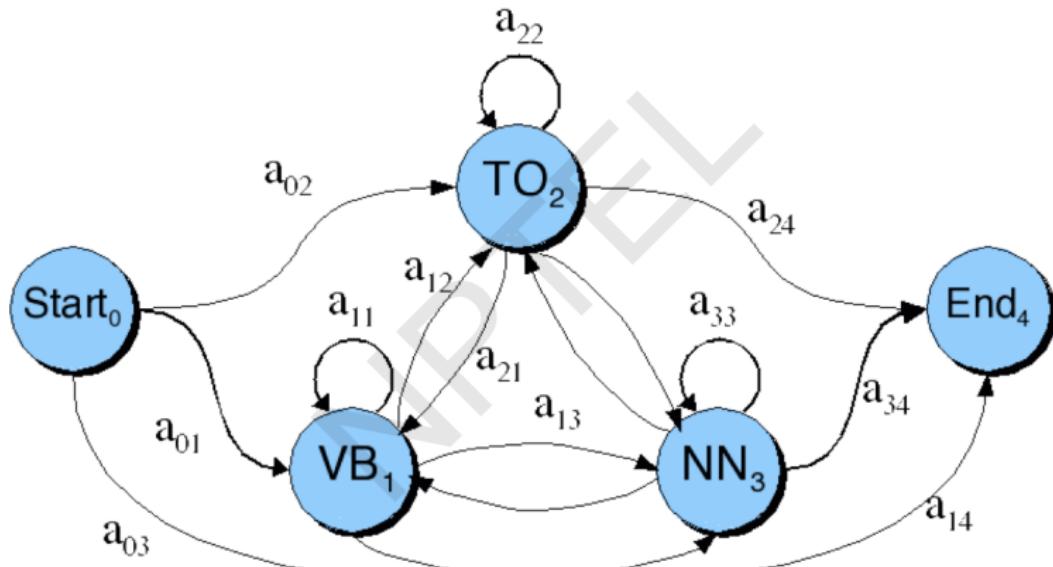
Hidden Markov Models (HMMs)

Elements of an HMM model

- A set of states (here: the tags)
- An output alphabet (here: words)
- Initial state (here: beginning of sentence)
- State transition probabilities (here $p(t_n|t_{n-1})$)
- Symbol emission probabilities (here $p(w_i|t_i)$)

Graphical Representation

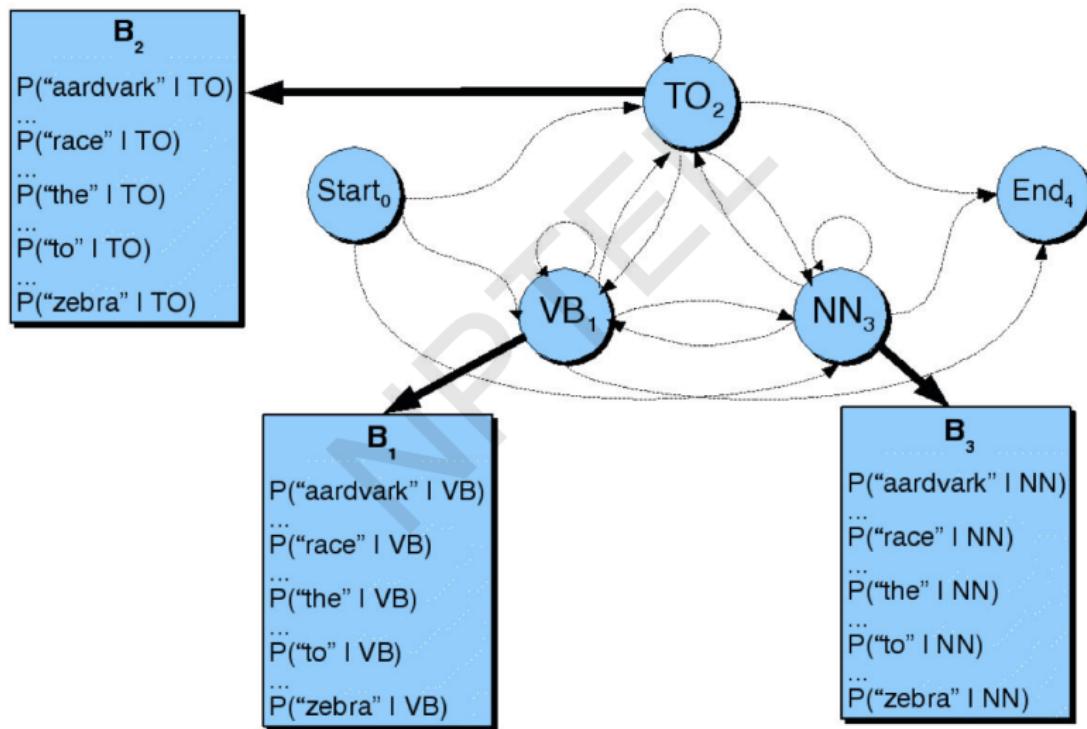
When tagging a sentence, we are walking through the state graph:



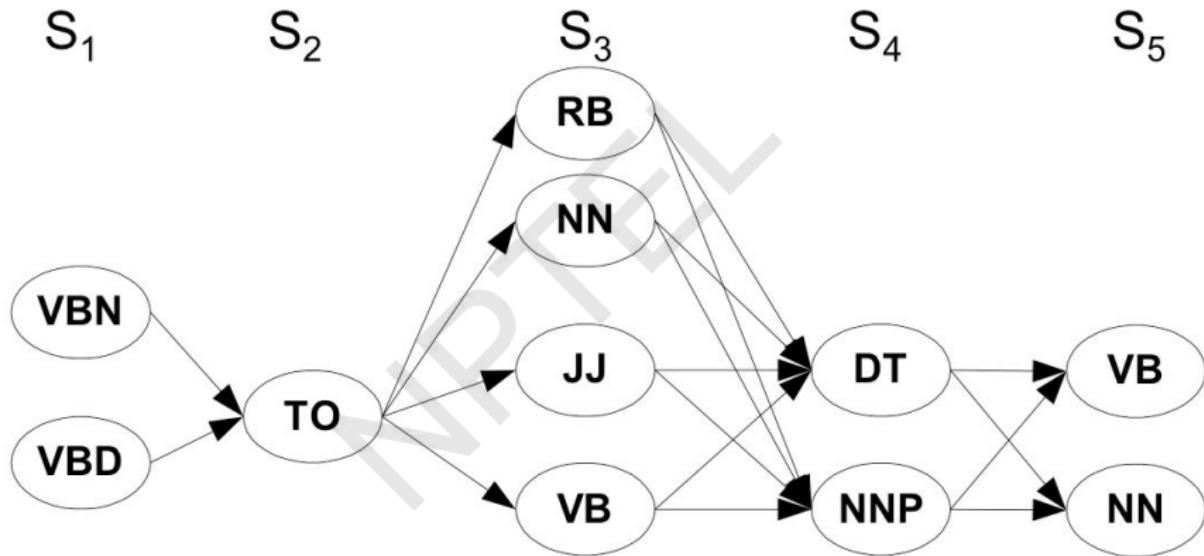
Edges are labeled with the state transition probabilities: $p(t_n|t_{n-1})$

Graphical Representation

At each state we emit a word: $P(w_n | t_n)$

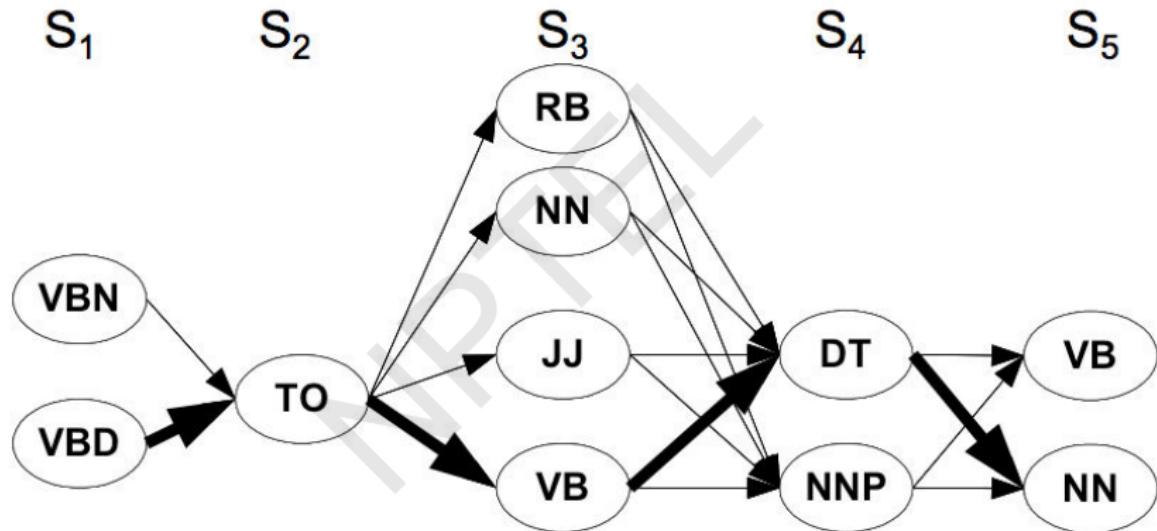


Walking through the states: best path



promised to back the bill

Walking through the states: best path



promised to back the bill