

AI6122 Text Data Management & Analysis

Topic: tokenization, morphology, stem,
lemmatization



What to be covered

- Token and Tokenization
- Stop words
- English Morphology
- Stemming and Lemmatization
- Edit Distance



Token

- A **token** is an instance of a sequence of characters in some document that are grouped together as a useful semantic unit for processing.
 - Often loosely referred to as words (or terms)
 - May include punctuations or emojis
- Tokenization: Identifying the tokens in a text that we may want to deal with
 - Also called word segmentation, word tokenization
 - Pretty much a prerequisite to doing anything interesting
- Example input: “*Friends, Romans and Countrymen*”
 - tokens: *Friends, Romans, Countrymen*
- Tokenization is language dependent, e.g., English, Chinese

Tokenization

- For English, why not just use white-space?
 - Mr. Sherwood said reaction to Sea Containers' proposal has been "very positive." In New York Stock Exchange composite trading yesterday, Sea Containers closed at \$62.625, up 62.5 cents.
 - "I said, 'what're you? Crazy?' " said Sadowsky. "I can't afford to do that."
- Using white-space gives you words like:
 - cents.
 - said,
 - positive."
 - Crazy?'



Tokenization: issues

- Word-internal punctuation
 - M.P.H. Ph.D. AT&T Google.com Yahoo!
- Other Examples:
 - Finland's capital → Finland? Finlands? Finland's?
 - Hewlett-Packard → Hewlett and Packard as two tokens?
 - state-of-the-art: break up hyphenated sequence.
 - co-education, lowercase, lower-case, lower case?
- How about names?
 - San Francisco: one token or two?
 - How do you decide it is one token? Then how about "San Francisco Airport"?
- What about numbers?
 - 3/20/91 Mar. 12, 1991 20/3/91
 - 55 B.C., B-52, My PGP key is 324a3df234cb23e
 - This is my number: (800) 234-2333



Tokenization: Implementation

- Tokenization needs to be run before any other language processing,
 - it needs to be very fast.
 - Standard method for tokenization is to use deterministic algorithms based on regular expressions compiled into very efficient finite state automata.
 - Example Python-based Natural Language Toolkit (NLTK)

```
>>> text = 'That U.S.A. poster-print costs $12.40...'
>>> pattern = r'''(?x)      # set flag to allow verbose regexps
...     ([A-Z]\.)+          # abbreviations, e.g. U.S.A.
...     | \w+(-\w+)*        # words with optional internal hyphens
...     | \$?\d+(\.\d+)?%?   # currency and percentages, e.g. $12.40, 82%
...     | \.\.\.            # ellipsis
...     | [][.,;'"'?():_-'] # these are separate tokens; includes ], [
...     '''
>>> nltk.regexp_tokenize(text, pattern)
['That', 'U.S.A.', 'poster-print', 'costs', '$12.40', '...']
```

Figure 2.11 A python trace of regular expression tokenization in the NLTK (Bird et al., 2009) Python-based natural language processing toolkit, commented for readability; the (?x) verbose flag tells Python to strip comments and whitespace. Figure from Chapter 3 of Bird et al. (2009).



Examples of other issues

- URL segmentation
 - www.dietsthatwork.com
 - www.choosespain.com
- Hashtag segmentation
 - #unitedbrokemyguitar
 - #manchesterunited
 - allows Twitter users to track what many people (especially people whom you aren't already following) are reporting or thinking about a particular topic or event.

What about code?

[Questions](#)[Tags](#)[Tour](#)[Users](#)

Stack Overflow is a question and answer site for professional and enthusiast programmers. It's 100% free, no registration required.

EF6 - Value cannot be null. Parameter name: extent

▲
0
▼
★
I'm using EF6 code first to create my db. Everything was working well last night, now when i run update-database command, I get the following exception:

```
PM> update-database
```

```
Specify the '-Verbose' flag to view the SQL statements being applied to the database.  
System.ArgumentNullException: Value cannot be null.  
Parameter name: extent
```

```
at System.Data.Entity.Utilities.Check.NotNull[T](T value, String parameterName) at  
at System.Data.Entity.Core.Mapping.StorageEntitySetMapping..ctor(EntityMappingType  
at System.Data.Entity.ModelConfiguration.Edm.DbDatabaseMappingExtensions.Create  
at System.Data.Entity.ModelConfiguration.Edm.Services.TableMappingGenerator  
at System.Data.Entity.ModelConfiguration.Edm.Services.DatabaseMappingService  
at System.Data.Entity.ModelConfiguration.Edm.Services.DatabaseMappingService  
at System.Data.Entity.DbModelBuilder.Build(DbProviderManifest providerManifest)  
at System.Data.Entity.DbModelBuilder.Build(DbConnection providerConnection)  
at System.Data.Entity.Internal.LazyInternalContext.CreateModel(LazyInternalContext  
at System.Data.Entity.Internal.RetryLazy`2.GetValue(TInput input)  
at System.Data.Entity.Internal.LazyInternalContext.InitializeContext()  
at System.Data.Entity.Internal.LazyInternalContext.get_CodeFirstModel()
```

[Run searchcode locally? Want to run your own version of searchcode](#)

About 419 results

[BinaryTree.java in rose_backup](#) https://github.com/dpick/rose_backup.git | 171 lines | Java

```
1. // BinaryTree class; stores a binary tree.  
2. //
```

```
8. // Also, the following tricky method:  
9. // void merge( Object root, BinaryTree t1, BinaryTree t2 )  
10. //      --> Construct a new tree
```

```
14. /**  
15. * BinaryTree class that illustrates the calling of BinaryNode recursive  
16. * routines and merge.
```

```
17. */  
18. public class BinaryTree {  
19.     /**
```



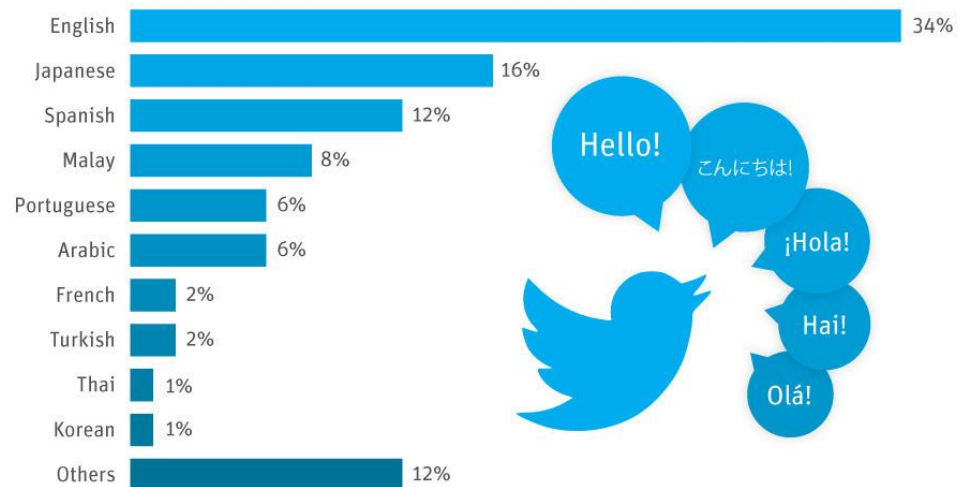
Is language really an issue here?



shd, ishould, shudd,
shuld, shoud, shud, shld,
sould, shouldd

Only 34% of All Tweets Are in English

Distribution of languages used in Tweets around the world (September 2013)



statista
The Statistics Portal

Mashable

Source: Semiocast

http://languagetime.files.wordpress.com/2013/12/131217_twitter_sprachen_mashable_n.jpg

Words grouping by Brown Clustering

^01110111001 (43)	everything everythin everthing evrything everythang everythingq everythng eveytthing everythinggg everyting evrythng errthang errything #everything errthing erything everythingggg errythang evrythin erthang lthing erthing everythingggg jony everythig everytin everyhing everithing everythign everythn everyhting everythingqq everythink erythang everything- everythingiing everythingggggg errrthang everythg everyword evreything everysong er'thing
^01110111010 (85)	nothing nothin nun nuthin nuttin nuffin 10x noting nthn nowt nuthn nuthing 100x nothingg nothn nothinq nutin notin nuffn nutn whatever #nodisrespect nuttn nothinggg #dontmeantobrag 1000x zilch nothinn #nothing nothng nutting nufin nuin nout nothinqq nthng nthing nothingggg nufn nofin nothen nthin nottin ntn nought ntg nothinnn nothign n0thing nothig
^011101110110 (108)	something somethin sumthin sumthing sumn something sth sumthn sumtin smth somthin suttin sumin somethingq summin treaters somethingg someting sumfin smthn somethn somethng summat smthng smthing sum'n sumthng smthg somn sumtn smething sometin somethign sum10 soemthing somethinn somethinggg something sumpin somin sumting something/someone somtin somehting somthn someshit sumptin something- smtg thereabouts
^011101110111 (36)	anything anythin nothing anythng anythingg anythingq anyting anythang anyth anytin anthing anythinggg anyfin vaart anythn anythign nethin nything anyhting #anything anything- anythg anythingggg nothing- anythink anythig anythingqq somethingggg nethng anyhing woodsen endsmeat anything/anyone aything anythingg anything

Source: http://www.cs.cmu.edu/~ark/TweetNLP/cluster_viewer.html



Tokenization: language issues

- French
 - **L'ensemble** → one token or two?
 - L ? L' ? Le ?
 - Want l'ensemble to match with un ensemble
- German noun compounds are not segmented
 - **Lebensversicherungsgesellschaftsangestellter**
 - ‘life insurance company employee’
 - German retrieval systems benefit greatly from a compound splitter module
 - Can give a 15% performance boost for German

Tokenization: language issues

- Chinese and Japanese have no spaces between words:
 - 莎拉波娃现在居住在美国东南部的佛罗里达。
 - Not always guaranteed a unique tokenization
- Japanese is more complicated, with multiple alphabets intermingled
 - Dates/amounts in multiple formats

フォーチュン500社は情報不足のため時間あた\$500K(約6,000万円)

Katakana

Hiragana

Kanji

Romaji

End-user can express query entirely in hiragana!

Tokenization: language issues

- Arabic (or Hebrew) is basically written right to left, but with certain items like numbers written left to right
- Words are separated, but letter forms within a word form complex ligatures

استقلت الجزائر في سنة 1962 بعد 132 عام من الاحتلال الفرنسي.

‘Algeria achieved its independence in 1962 after 132 years of French occupation.’

- With Unicode, the surface presentation is complex, but the stored form is straightforward

Segmentation in Chinese

- Words composed of characters; average word is 2.4 characters long.
- A simple segmentation algorithm based on dictionary:
 - Maximum Matching or Maxmatch
 - Given a lexicon of Chinese, and a string
 - Start a pointer at the beginning of the string
 - Find the longest word in dictionary that matches the string starting at pointer
 - Move the pointer over the word in string
 - Go to 2
- State-of-the-art solutions are mostly **probabilistic** and deep learning
 - http://nlpprogress.com/chinese/chinese_word_segmentation.html
 - <https://github.com/topics/chinese-text-segmentation>
- Recent study: for most Chinese NLP tasks, take characters rather than words as input works better.
 - Characters are at a reasonable semantic level for most applications,
 - Most word standards result in a huge vocabulary with large numbers of very rare words



Maximum Matching Word Segmentation

Lexicon (Dictionary)
the
table
down
there
...

thetabledownthere



the table down there

Lexicon (Dictionary)
the
theta
table
down
there
bled
own

thetabledownthere



theta bled own there



Byte-Pair Encoding for (subword) Tokenization

- Data driven, and not confine to words or characters
 - Resultant **token** could be smaller than words e.g., -est, er,
 - Or larger than words, e.g., New York Times
 - Have **subword** tokens to deal with unknown words.
 - Proposed for machine translation, dealing with rare or new words
- Given training data (words are annotated)
 - BPE starts with the set of symbols equal to the set of characters
 - Each word is represented as a sequence of characters plus a special end-of-word symbol _
 - At each step, the most frequent pair ('A', 'B') of symbol pairs is merged, and replaced with the newly merged symbol ('AB')
 - Repeat the step, until we have done k merges (k is a parameter)
 - The final symbol vocabulary size is equal to the size of the initial vocabulary, plus the number of merge operations



BEP Example

- BPE runs inside words, we don't merge across word boundaries
 - The input is a dictionary of words together with their counts

	dictionary	vocabulary
5	l o w _	_, d, e, i, l, n, o, r, s, t, w
2	l o w e s t _	
6	n e w e r _	
3	w i d e r _	
2	n e w _	

- Count symbol pairs: r _ has highest frequency 9 (6 in newer_ and 3 in wider_)
- Then r_ is merged



After merging “r _” and then “e r _”

dictionary

5 l o w _
2 l o w e s t _
6 n e w e r_
3 w i d e r_
2 n e w _

vocabulary

, d, e, i, l, n, o, r, s, t, w, r

dictionary

5 l o w _
2 l o w e s t _
6 n e w e r_
3 w i d e r_
2 n e w _

vocabulary

, d, e, i, l, n, o, r, s, t, w, r, e r_



Continue merging

dictionary

5 l o w _
 2 l o w e s t _
 6 n ew er_
 3 w i d er_
 2 n ew _

vocabulary

, d, e, i, l, n, o, r, s, t, w, r, er_, ew

Merge

Current Vocabulary

(n, ew) _, d, e, i, l, n, o, r, s, t, w, r_, er_, ew, new
 (l, o) _, d, e, i, l, n, o, r, s, t, w, r_, er_, ew, new, lo
 (lo, w) _, d, e, i, l, n, o, r, s, t, w, r_, er_, ew, new, lo, low
 (new, er_) _, d, e, i, l, n, o, r, s, t, w, r_, er_, ew, new, lo, low, newer_
 (low, _) _, d, e, i, l, n, o, r, s, t, w, r_, er_, ew, new, lo, low, newer_, low_



Apply BEP for (subword) tokenization

- The merging sequence is what we have learned from training data
- To apply BEP for tokenization on test data
 - We run the merges we have learned, greedily, in the order we learned them
 - The frequencies in the test data don't play a role, just the frequencies in the training data.
- Follow the example:
 - Segment each test sentence word into characters.
 - Apply the first rule: replace every instance of “r _” in the test corpus with “r_”, and then the second rule: replace every instance of “e r_” in the test corpus with “er_”, and so on.
 - By the end, word “n e w e r” , it would be tokenized as a full word.
 - a new (unknown) word like “l o w e r” would be merged into the two tokens “low” and “er”



Sentence Segmentation

- !, ? are relatively unambiguous
- Period “.” is quite ambiguous
 - Sentence boundary
 - Abbreviations like Inc. or Dr.
- General idea:
 - Build a binary classifier:
 - Looks at a “.”
 - Decides EndOfSentence/NotEOS
 - Could be hand-written rules, sequences of regular expressions, or machine-learning



Now we mainly focus on words

- Stop words
- English Morphology
- Stemming and Lemmatization



Stop words

- With a stop list, you exclude from the dictionary entirely the commonest words. :
 - They have little semantic content: the, a, and, to, be
 - There are a lot of them: ~30% of postings for top 30 words
- But the trend is away from doing this:
 - Good compression techniques means the space for including stopwords in a system is very small
 - Good query optimization techniques mean you pay little at query time for including stop words.
 - You need them for:
 - Phrase queries: “King of Denmark”
 - Various song titles, etc.: “Let it be”, “To be or not to be”
 - “Relational” queries: “flights to London”

English Morphology

- Morphology is the study of the ways that words are built up from smaller meaningful units called **morphemes**
- We can usefully divide morphemes into two categories
 - **Stems**: The core meaning-bearing units
 - **Affixes**: Bits and pieces that adhere to stems to change their meanings and grammatical functions
 - Example
 - cat → cats
 - regular → irregular



English Morphology

- We can further divide morphology up into two broad classes
 - **Inflectional:** has **the same word class** as the original, cat -> cats
 - **Derivational:** changes of **word class**, care -> careless
- Word Classes
 - By word class, we have in mind familiar notions like noun and verb
 - We'll go into the details in POS tagging
 - Right now we're concerned with word classes because **the way that stems and affixes combine** is based to a large degree on the word class of the stem

Inflectional Morphology

- **Inflectional morphology** concerns the combination of stems and affixes where the resulting word:
 - Has **the same word class** as the original
 - Fill some syntactic function like agreement
- Nouns are simple
 - Markers for plural and possessive
 - Example: table, tables; children, children's
- Verbs are only slightly more complex
 - Markers appropriate to the tense of the verb
 - Example: walk, walks, walking



Regulars and Irregulars

- It is a little complicated by the fact that some words misbehave (refuse to follow the rules)
 - Mouse/mice, goose/geese, ox/oxen
 - Go/went, fly/flew
- The terms **regular** and **irregular** are used to refer to words that follow the rules and those that don't



Regular and Irregular Verbs

- Inflectional morphology in English is fairly straightforward
- But is complicated by the fact that there are irregularities
- Regulars...
 - Walk, walks, walking, walked, walked
- Irregulars
 - Catch, catches, catching, **caught**, **caught**
 - Cut, cuts, cutting, **cut**, **cut**



Derivational Morphology

- More complicated.
- Many paths are possible...
 - Start with **compute**
 - **Computer** -> **computerize** -> **computerization**
 - **Computer** -> **computerize** -> **computerizable**
- Meaning change
 - E.g., **care** -- **careless**
- Changes of **word class**



Derivational Examples

- Nouns and Verbs → Adjectives

-al	computation	computational
-able	embrace	embraceable
-less	clue	clueless

- Verbs and Adjectives → Nouns

-ation	computerize	computerization
-ee	appoint	appointee
-er	kill	killer
-ness	fuzzy	fuzziness

Normalization

- Need to “normalize” terms
 - Information Retrieval: indexed text & query terms must have same form.
 - We want to match **U.S.A.** and **USA**
- We implicitly define equivalence classes of terms
 - e.g., deleting periods in a term
- Alternative: asymmetric expansion:
 - Enter: **window** Search: **window, windows**
 - Enter: **windows** Search: **Windows, windows, window**
 - Enter: **Windows** Search: **Windows**
- Potentially more powerful, but less efficient

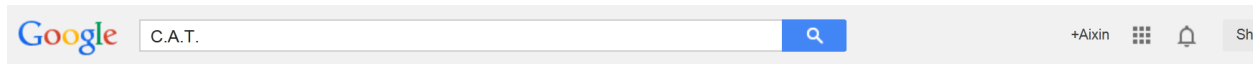


Case folding

- Applications like IR: reduce all letters to lower case
 - Since users tend to use lower case
 - Possible exception: upper case in mid-sentence?
 - e.g., **General Motors**
 - **Fed** vs. **fed**
 - **SAIL** vs. **sail**
- For sentiment analysis, machine translation, information extraction
 - **Case is helpful** (US versus us is important)



Google's answer to query "C.A.T."



Web Images Maps News More Search tools

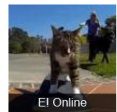
About 461,000,000 results (0.25 seconds)

Cat - Wikipedia, the free encyclopedia

en.wikipedia.org/wiki/Cat

The domestic cat (*Felis catus* or *Felis silvestris catus*) is a small, usually furry, domesticated, and carnivorous mammal. It is often called the housecat when kept ...
[African wildcat](#) - [Creme Puff](#) - [List of cat breeds](#) - [Cats and humans](#)

News for C.A.T.



Is this the coolest cat on the planet? Fun-loving feline rides a skateboard

Metro - 4 hours ago

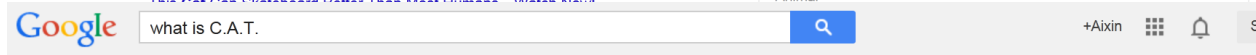
Sleeping all day doesn't appear to appeal to this fun-loving feline who joined his owner for a skateboard adventure around a beautiful beach ...



More images

Cat

Animal



Web Images Maps More Search tools

About 1,340,000,000 results (0.41 seconds)

Common Admission Test - Wikipedia, the free encyclopedia

en.wikipedia.org/wiki/Common_Admission_Test

The Common Admission Test (CAT) is a computer based test held in India. This test scores a person on the bases of quantitative ability, data interpretation, ...

[History](#) - [Approval](#) - [Exam format](#) - [See also](#)

Cat - Wikipedia, the free encyclopedia

en.wikipedia.org/wiki/Cat

The domestic cat (*Felis catus* or *Felis silvestris catus*) is a small, usually furry, domesticated, and carnivorous mammal. It is often called the housecat when kept ...

Methcathinone - Wikipedia, the free encyclopedia

en.wikipedia.org/wiki/Methcathinone

Methcathinone (α -methylamino-propiofenone or ephedrone) (sometimes called "cat" or "jeff") is a monoamine alkaloid and psychoactive stimulant similar to ...

What is CAT Qualification - OpenTuition

opentuition.com/fia/what-is-cat-qualification/

CAT stands for Certified Accounting Technician and the qualification provides a strong foundation of knowledge in finance and accounting. It is provided by the ...

See results about

Common Admission Test

The Common Admission Test is a computer based test held in India. This test scores a person on the ...

Cat

Animal

The domestic cat is a small, usually furry, domesticated, and carnivorous mammal. It is often ...



Feedback/More info

Category filter:

Acronym Definition

CAT	Category
CAT	Catalog
CAT	Convention Against Torture (<i>United Nations</i> ;
CAT	Computerized Axial Tomography (<i>medical in</i>
CAT	Centre for Alternative Technology (<i>Machynlle</i>
CAT	Catalog (<i>File Name Extension</i>)
CAT	Common Admission Test (<i>India</i>)
CAT	Caterpillar, Inc.
CAT	Community Action Team
CAT	Catalunya
CAT	Catalyst
CAT	Computer Aided Translation
CAT	Catastrophe (<i>Claims Office</i>)
CAT	Catamaran
CAT	Computer-Assisted Translation
CAT	Certified Accounting Technician
CAT	Catastrophic
CAT	Call Any Time
CAT	Catalan Language (<i>ISO 639-2 Code</i>)
CAT	Cataract (<i>ophthalmology</i>)
CAT	Catalytic Converter (<i>automobiles</i>)
CAT	Catapult
CAT	Catalonia (<i>country identifier</i>)
CAT	Centre d'Aide par le Travail (<i>French: Center</i> ;
CAT	Catechism
CAT	Computerized Adaptive Testing
CAT	Computer Adaptive Test
CAT	Civil Air Transport (<i>CIA front airline during 1</i>
CAT	California Achievement Test
CAT	Concatenate
CAT	Capital Area Transit (<i>Raleigh, NC</i>)
CAT	Catalase
CAT	Chloramphenicol Acetyltransferase



Lemmatization

- Reduce inflections or variant forms to base form
 - am, are, is → be
 - car, cars, car's, cars' → car
- the boy's cars are different colors →
 - the boy car be different color
- Lemmatization: have to find **correct dictionary headword form**
 - Determining that two words have the same root, despite their surface differences



Light-Weight Morphology

- Sometimes you just need to know the stem of a word and you don't care about the structure.
 - E.g. camera, cameras
- In fact you may not even care if you get the right stem, as long as you get a consistent string—**Stemming**
- Stemming for Information Retrieval
 - Run a stemmer on the documents to be indexed
 - Run a stemmer on users' queries
 - Match to the index



Porter's stemming

- No lexicon needed
- Basically a set of staged sets of rewrite rules that strip suffixes
 - ING $\rightarrow \epsilon$ (e.g., monitoring \rightarrow monitor)
 - SSES \rightarrow SS (e.g., grasses \rightarrow grass)
- More Example (recursive)
 - Computerization
 - ization \rightarrow -ize computerize
 - ize $\rightarrow \epsilon$ computer



Porter's stemming

- Handles both inflectional and derivational suffixes
- Doesn't guarantee that the resulting stem is really a stem
 - Lack of guarantee doesn't matter for IR
- Code:
 - <http://tartarus.org/martin/PorterStemmer/>
 - Implementations in C, Java, Perl, Python, C#, Lisp, Ruby, VB, javascript, php, Prolog, Haskell, matlab, tcl, D, and erlang



Stemming used in search

- Recall: reduce false negative? (Not matching things that we should have matched)
 - Query: “dog”
 - Doc 1: I love my dog
 - Doc 2: I do not like dogs
- Precision: increase false positives? (Matching strings that we should not have matched)
 - Query: “policy”
 - Doc 3: Singapore policy on gum
 - Doc 4: Singapore police cool

Works in this case

Wrong results here

policy—policies
police—policing

Stemming

- Reduce terms to their stems in information retrieval
- *Stemming* is **crude chopping of affixes**
 - language dependent
 - e.g., ***automate(s), automatic, automation*** all reduced to ***automat***.

*for example compressed
and compression are both
accepted as equivalent to
compress.*



for exampl compress and
compress ar both accept
as equal to compress



Stemmer vs Dictionary

- Stemming Rules more efficient than a dictionary
 - Algorithmic stemmers can be fast (and lean): 1 Million words in 6 seconds on 500 MHz PC
- No maintenance even if things change
- Better to ignore irregular forms (exceptions) than to complicate the algorithm
 - not much lost in practice
 - 80/20 Rule



Other stemmers

- Other stemmers exist, e.g., Lovins stemmer
 - <http://www.comp.lancs.ac.uk/computing/research/stemming/general/lovins.htm>
 - Single-pass, longest suffix removal (about 250 rules)
- Full morphological analysis – at most modest benefits for retrieval
- Do stemming and other normalizations help?
 - English: very mixed results. Helps recall for some queries but harms precision on others
 - E.g., operative (dentistry) \Rightarrow oper
 - Definitely useful for Spanish, German, Finnish, ...
 - 30% performance gains for Finnish!

Language-specificity

- Many of the above features embody transformations that are
 - Language-specific and
 - Often, application-specific
- These are “plug-in” addenda to the indexing process
- Both open source and commercial plug-ins are available for handling these

Normalization: other languages

- Accents: e.g., French résumé vs. resume.
- Umlauts: e.g., German: Tuebingen vs. Tübingen
 - Should be equivalent
- Most important criterion:
 - How are **your users like to write** their queries for these words?
- Even in languages that standardly have accents, users often may not type them
 - Often best to normalize to a de-accented term
 - Tuebingen, Tübingen, Tübingen → Tübingen

Normalization: other languages

- Normalization of things like date forms
 - 7月30日 vs. 7/30
 - Japanese use of kana vs. Chinese characters
- Tokenization and normalization may depend on the language and so is intertwined with language detection
- Crucial: Need to “normalize” indexed text and query terms into **the same form**

Morgen will ich in MIT ...

Is this
German “mit”?

Thesauri and soundex

- Do we handle synonyms and homonyms?
 - E.g., by hand-constructed equivalence classes
 - ***car = automobile color = colour***
 - We can rewrite to form equivalence-class terms
 - When the document contains ***automobile***, index it under ***car-automobile*** (and vice-versa)
 - Or we can expand a query
 - When the query contains ***automobile***, look under ***car*** as well
- What about spelling mistakes?
 - One approach is soundex, which forms equivalence classes of words based on phonetic heuristics

Basic concepts: Phrase Detection

- Important for English
 - New York City Police Department
 - Bill Gates spoke on the benefits of Windows
- Essential for CJK (Chinese, Japanese, Korean)
 - 新加坡是个美丽的城市 [Singapore is a beautiful city]
- Approaches
 - Dictionary Based
 - Most Accurate; Needs maintenance (by humans)
 - Learnt/Extracted from Corpus
 - Hidden Markov Model; N-Grams; Statistical Analysis
 - Suffix Tree Phrase Detection (via statistical counting)

Example of Extracted Phrases on 20 Newsgroup dataset

south
south africa
south africa internet
south africa po
south african
south african government
south african intelligence
south african libertarian
south america
south atlantic
south dakota
south dakota writes
south georgia
south georgia island
south pacific
south pacific island

san diego
san diego ca
san francisco
san francisco bay
san francisco chronicle
san francisco giants
san francisco police
san francisco police inspector
san francisco police inspector ron
san francisco police intelligence
san francisco police intelligence unit
san francisco police officer
san jose
san jose ca
san jose mercury

high
high density
high end
high enough
high frequency
high hockey
high just
high level
high performance
high power
high quality
high ranking
high ranking crime
high ranking initiate
high resolution
high school
high school students
high speed
high speed collision
high sticking
high tech
high voltage
highend
higher

united
united kingdom
united nations
united nations quite
united states
united states attempt
united states code
united states government
united states holocaust
united states officially
united states senate

Edit Distance

- The minimum edit distance between two strings is the minimum number of editing operations needed to transform one into the other
 - Insertion
 - Deletion
 - Substitution
- Spell correction
 - The user typed “**graffe**”
 - Which is closest?
 - graf
 - graft
 - grail
 - giraffe



Minimum Edit Distance

- Two strings and their **alignment**:

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				

- If each operation has cost of 1
 - Distance between these is 5
- If substitutions cost 2 (Levenshtein)
 - Distance between them is 8



Other uses of Edit Distance in NLP

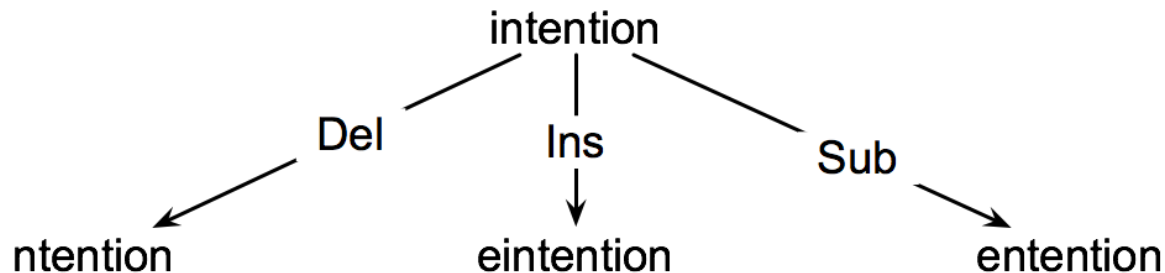
- Evaluating Machine Translation and speech recognition
 - Spokesman **confirms** senior **government** adviser was shot
 - Spokesman **said** **the** senior adviser was shot **dead**

S I D I
- Named Entity Extraction and Entity Coreference
 - **IBM Inc.** announced today
 - **IBM** profits
 - **Stanford President John Hennessy** announced yesterday
 - for **Stanford University President John Hennessy**



How to find the Min Edit Distance?

- Searching for a path (sequence of edits) from the start string to the final string:
 - Initial state: the word we're transforming
 - Operators: insert, delete, substitute
 - Goal state: the word we're trying to get to
 - Path cost: what we want to minimize: the number of edits



Minimum Edit as Search

- But the space of all edit sequences is huge!
 - We can't afford to navigate naïvely
 - Lots of distinct paths wind up at the same state.
 - We don't have to keep track of all of them
 - Just the shortest path to each of those re-visited states.

i n t e n t i o n	← <i>delete i</i>
n t e n t i o n	← <i>substitute n by e</i>
e t e n t i o n	← <i>substitute t by x</i>
e x e n t i o n	← <i>insert u</i>
e x e n u t i o n	← <i>substitute n by c</i>
e x e c u t i o n	

Defining Min Edit Distance

- For two strings
 - X of length n
 - Y of length m
- We define $D(i, j)$ to be 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
 - The edit distance between X and Y is thus $D(n, m)$



Dynamic Programming for Minimum Edit Distance

- Dynamic programming: A tabular computation of $D(n, m)$
 - Solving problems by combining solutions to subproblems.
- Bottom-up
 - We compute $D(i, j)$ for small i, j , then compute larger $D(i, j)$ based on previously computed smaller values
 - That is: we compute $D(i, j)$ for all i ($0 < i < n$) and j ($0 < j < m$)



Defining Min Edit Distance (Levenshtein)

- Initialization

- $D(i, 0) = i$

- $D(0, j) = j$

- Recurrence Relation:

- For each $i = 1 \dots M$

- For each $j = 1 \dots N$

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

- Termination:

- $D(N, M)$ is distance

$$D[i, j] = \min \begin{cases} D[i-1, j] + 1 \\ D[i, j-1] + 1 \\ D[i-1, j-1] + \begin{cases} 2; & \text{if } \text{source}[i] \neq \text{target}[j] \\ 0; & \text{if } \text{source}[i] = \text{target}[j] \end{cases} \end{cases}$$




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

$$D[i, j] = \min \begin{cases} D[i-1, j] + 1 \\ D[i, j-1] + 1 \\ D[i-1, j-1] + \begin{cases} 2; & \text{if } source[i] \neq target[j] \\ 0; & \text{if } source[i] = target[j] \end{cases} \end{cases}$$




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

- Edit distance isn't sufficient
 - We often need to **align** each character 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



MinEdit 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	



Result of Backtrace

- Two strings and their **alignment**:

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



Topics covered

- Token and Tokenization
- Stop words
- English Morphology
- Stemming and Lemmatization
- Edit Distance

