GENERAL ASSEMBLY

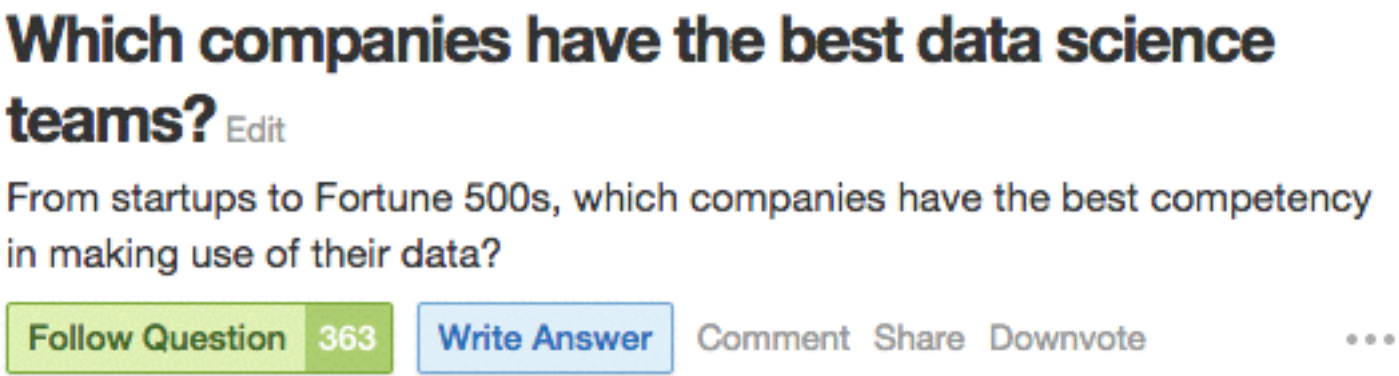
INTRO TO DATA SCIENCE

LECTURE 16: TEXT MINING

Francesco Mosconi
DAT10 SF // November 23, 2014

DATA SCIENCE IN THE NEWS

DATA SCIENCE IN THE NEWS



source: http://www.quora.com/Which-companies-have-the-best-data-science-teams

DATA SCIENCE IN THE NEWS

source: http://blog.datacamp.com/wp-content/uploads/2014/08/How-to-become-a-data-scientist.jpg

LAST TIME

I. RECAP OF PROBABILITY
II. NAÏVE BAYESIAN CLASSIFICATION

QUESTIONS?

AGENDA

I. QUICK REVIEW OF REGULAR EXPRESSIONS
II. DEFINITIONS OF TERMS IN NLP

I. REGULAR EXPRESSIONS

Q: What are regular expressions?

Q: What are regular expressions?
A: a regular expression (abbreviated regex or regexp) is a sequence of characters that forms a search pattern, mainly for use in pattern matching, i.e., “find and replace’-like operations.

Q: What are regular expressions?
A: a regular expression (abbreviated regex or regexp) is a sequence of characters that forms a search pattern, mainly for use in pattern matching, i.e., “find and replace’-like operations.

The concept arose in the 1950’s, when the American mathematician Stephan Kleene formalized the description of a regular language, and came into common use with the unix text processing utilities ed, an editor, and grep (global regular expression print), a filter.

Each character in a regular expression is either understood to be a metacharacter with its special meaning, or a regular character with its literal meaning.

Each character in a regular expression is either understood to be a metacharacter with its special meaning, or a regular character with its literal meaning.

Together, they can be used to identify textual material of a given pattern, or process a number of instances of it that can vary from a precise equality to a very general similarity of the pattern.

Each character in a regular expression is either understood to be a metacharacter with its special meaning, or a regular character with its literal meaning.

Together, they can be used to identify textual material of a given pattern, or process a number of instances of it that can vary from a precise equality to a very general similarity of the pattern.

The pattern sequence itself is an expression that is a statement in a language designed specifically to represent prescribed targets in the most concise and flexible way to direct the automation of text processing of general text files, specific textual forms, or of random input strings.

A very simple use of a regular expression would be to locate the same word spelled two different ways in a text editor, for example seriali**[**sz**]**e.

A very simple use of a regular expression would be to locate the same word spelled two different ways in a text editor, for example seriali**[**sz**]**e.

A wildcard match can also achieve this, but wildcard matches differ from regular expressions in that wildcards are limited to what they can pattern (having fewer metacharacters and a simple language-base), whereas regular expressions are not.

A very simple use of a regular expression would be to locate the same word spelled two different ways in a text editor, for example seriali**[**sz**]**e.

A wildcard match can also achieve this, but wildcard matches differ from regular expressions in that wildcards are limited to what they can pattern (having fewer metacharacters and a simple language-base), whereas regular expressions are not.

A usual context of wildcard characters is in globbing similar names in a list of files, whereas regular expressions are usually employed in applications that pattern-match text strings in general.

A very simple use of a regular expression would be to locate the same word spelled two different ways in a text editor, for example seriali**[**sz**]**e.

A wildcard match can also achieve this, but wildcard matches differ from regular expressions in that wildcards are limited to what they can pattern (having fewer metacharacters and a simple language-base), whereas regular expressions are not.

A usual context of wildcard characters is in globbing similar names in a list of files, whereas regular expressions are usually employed in applications that pattern-match text strings in general.

For example, the simple regexp**^[** \t**]**+**|[** \t**]**+**$** matches excess whitespace at the beginning and end of a line. An advanced regexp used to match any numeral is **^[**+-**]**?**(**\d+ \.?\d*|\.\d+**)([**eE**][**+-**]**?\d+**)**?**$**

The first metacharacters we'll look at are [and]. They're used for specifying a character class, which is a set of characters that you wish to match. Characters can be listed individually, or a range of characters can be indicated by giving two characters and separating them by a '-'.

The first metacharacters we'll look at are [and]. They're used for specifying a character class, which is a set of characters that you wish to match. Characters can be listed individually, or a range of characters can be indicated by giving two characters and separating them by a '-'.

For example, [abc] will match any of the characters a, b, or c; this is the same as [a-c], which uses a range to express the same set of characters. If you wanted to match only lowercase letters, your RE would be [a-z].

\d
Matches any decimal digit; this is equivalent to the class [0-9].
\D
Matches any non-digit character; this is equivalent to the class [^0-9].
\s
Matches any whitespace character; this is equivalent to the class [\t\n\r\f\v].
\S
Matches any non-whitespace character; this is equivalent to the class [^ \t\n\r\f\v].
\w
Matches any alphanumeric character; this is equivalent to the class [a-zA-Z0-9_].
\W
Matches any non-alphanumeric character; this is equivalent to the class [^a-zA-Z0-9_].

Performing Matches

Method/Attribute	Purpose
`match()`	Determine if the RE matches at the beginning of the string.
`search()`	Scan through a string, looking for any location where this RE matches.
`findall()`	Find all substrings where the RE matches, and returns them as a list.
`finditer()`	Find all substrings where the RE matches, and returns them as an *iterator*.

Querying the match object

Method/Attribute	Purpose
`group()`	Return the string matched by the RE
`start()`	Return the starting position of the match
`end()`	Return the ending position of the match
`span()`	Return a tuple containing the (start, end) positions of the match

With that, you are now prepared for…

With that, you are now prepared for…incredibly nerdy humor:

Source: http://xkcd.com/1313/

II. NATURAL LANGUAGE PROCESSING

Q: What is natural language processing?

A: Natural language processing (NLP) is a field of computer science, artificial intelligence, and linguistics concerned with the interactions between computers and human (natural) languages.

Q: What is natural language processing?

A: Natural language processing (NLP) is a field of computer science, artificial intelligence, and linguistics concerned with the interactions between computers and human (natural) languages.

As such, NLP is related to the area of human–computer interaction (HCI)

Q: What is natural language processing?

A: Natural language processing (NLP) is a field of computer science, artificial intelligence, and linguistics concerned with the interactions between computers and human (natural) languages.

As such, NLP is related to the area of human–computer interaction (HCI).

Many challenges in NLP involve natural language understanding -- that is, enabling computers to derive meaning from human or natural language input.

Q: How do we derive meaning from a stream of text?

Q: How do we derive meaning from a stream of text?

A: we break it up into smaller units and build meaning from them

Q: what is language made of?

Q: what is language made of?

A: words!

Q: what is language made of?

A: words!

And words are defined by “Word Boundaries”

Q: What are word boundaries?

Q: What are word boundaries?

A: The task of defining what constitutes a "word" involves determining where one word ends and another word begins. In other words, identifying word boundaries.

Q: What are word boundaries?

A: The task of defining what constitutes a "word" involves determining where one word ends and another word begins. In other words, identifying word boundaries.

This process can be generalized into what's called Tokenization

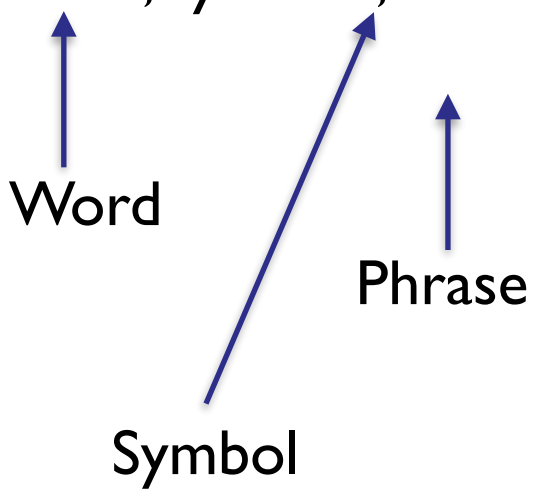
Q: What is Tokenization?

Q: What is Tokenization?

A: Tokenization is the process of breaking a stream of text up into words, phrases, symbols, or other meaningful elements called tokens.

Q: What is Tokenization?

A: Tokenization is the process of breaking a stream of text up into words, phrases, symbols, or other meaningful elements called tokens.



Q: What is Tokenization?

A: Tokenization is the process of breaking a stream of text up into words, phrases, symbols, or other meaningful elements called tokens.

The list of tokens becomes **input** for further processing such as parsing or text mining. Tokenization is useful both in linguistics (where it is a form of text segmentation), and in computer science, where it forms part of lexical analysis.

Q: What do these words have in common?

fish
fisher
fisherman
fishy
fishing
fished

Q: What do these words have in common?

fish
fisher
fisherman
fishy
fishing
fished

The stem

Q: What is stemming?

Q: What is stemming?

A: In linguistic morphology and information retrieval, stemming is the process for reducing inflected (or sometimes derived) words to their stem, base or root form—generally a written word form.

The Stem

need not be identical to the morphological root of the word; *it is usually sufficient that related words map to the same stem, even if this stem is not in itself a valid root.*

The Stem

need not be identical to the morphological root of the word; *it is usually sufficient that related words map to the same stem, even if this stem is not in itself a valid root.*

Algorithms for stemming have been studied in computer science since the 1960s. Many search engines treat words with the same stem as synonyms as a kind of query expansion, a process called **conflation**.

Ok, so we split a text into tokens, we traced the root of each token through stemming… now what?

Ok, so we split a text into tokens, we traced the root of each token through stemming… now what?

Now we need to translate this into maths and stats

A common tool is **tf-idf**

Q: What is tf-idf?

Q: What is tf-idf?

A: term frequency–inverse document frequency

Q: What is tf-idf?

A: term frequency–inverse document frequency:

a numerical statistic which reflects how important a word is to a document in a collection or corpus.

Q: What is tf-idf?

A: term frequency–inverse document frequency:

a numerical statistic which reflects how important a word is to a document in a collection or corpus.

Often used as a weighting factor in information retrieval and text mining. The tf-idf value increases proportionally to the number of times a word appears in the document, but is offset by the frequency of the word in the corpus, which helps to control for the fact that some words are generally more common than others.

Q: How does tf-idf work?

Q: How does tf-idf work?

A: its value increases proportionally to the number of times a word appears in the document, but is offset by the frequency of the word in the corpus, which helps to control for the fact that some words are generally more common than others.

Variations of the tf–idf weighting scheme are often used by search engines as a central tool in scoring and ranking a document's relevance given a user query.

tf–idf can be successfully used for stop-words filtering in various subject fields including text summarization and classification. One of the simplest ranking functions is computed by summing the tf–idf for each query term; many more sophisticated ranking functions are variants of this simple model.

Q: why is tf-idf important

A: because it's a way to go from a text to a vector in multi dimensional space

TEXT =========> VECTOR

Q: why is tf-idf important

A: because it's a way to go from a text to a vector in multi dimensional space

TEXT =========> VECTOR

And now I can apply all the usual tricks….

TEXT ========> VECTOR

And now I can apply all the usual tricks….
….like for example cosine similarity

Q: What is cosine similarity?

A: Cosine similarity is a measure of similarity between two vectors of an inner product space that measures the cosine of the angle between them. The cosine of 0° is 1, and it is less than 1 for any other angle. It is thus a judgement of orientation and not magnitude: two vectors with the same orientation have a Cosine similarity of 1, two vectors at 90° have a similarity of 0, and two vectors diametrically opposed have a similarity of -1, independent of their magnitude. Cosine similarity is particularly used in positive space, where the outcome is neatly bounded in [0,1].

Note that these bounds apply for any number of dimensions, and Cosine similarity is most commonly used in high-dimensional positive spaces. For example, in Information Retrieval and text mining, each term is notionally assigned a different dimension and a document is characterized by a vector where the value of each dimension corresponds to the number of times that term appears in the document. Cosine similarity then gives a useful measure of how similar two documents are likely to be in terms of their subject matter. The technique is also used to measure cohesion within clusters in the field of data mining.

Given two vectors of attributes, A and B, the cosine similarity , cos(theta) is the dot product of A and B over the product of the magnitudes of A and B:

$$\text{similarity} = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} (A_i)^2} \times \sqrt{\sum_{i=1}^{n} (B_i)^2}}$$

The resulting similarity ranges from −1 meaning exactly opposite, to 1 meaning exactly the same, with 0 usually indicating independence, and in-between values indicating intermediate similarity or dissimilarity.

For text matching, the attribute vectors *A* and *B* are usually the term frequency vectors of the documents. The cosine similarity can be seen as a method of normalizing document length during comparison.

In the case of information retrieval, the cosine similarity of two documents will range from 0 to 1, since the term frequencies (tf-idf weights) cannot be negative. The angle between two term frequency vectors cannot be greater than 90°.

TEXT ========> VECTOR

And now I can apply all the usual tricks….
….like for example singular value decomposition

Q: What is latent semantic analysis?

Q: What is latent semantic analysis?

A: Latent semantic analysis (LSA) is a technique in natural language processing, in particular in vectorial semantics, of analyzing relationships between a set of documents and the terms they contain by producing a set of concepts related to the documents and terms. LSA assumes that words that are close in meaning will occur in similar pieces of text. A matrix containing word counts per paragraph (rows represent unique words and columns represent each paragraph) is constructed from a large piece of text and a mathematical technique called singular value decomposition (SVD) is used to reduce the number of columns while preserving the similarity structure among rows. Words are then compared by taking the cosine of the angle between the two vectors formed by any two rows. Values close to 1 represent very similar words while values close to 0 represent very dissimilar words.

III. NATURAL LANGUAGE PROCESSING - OTHER TOOLS

Other tools for NLP

- word sense disambiguation
- topic models
- parts of speech
- Flesch-Kincaid index
- Latent Dirichlet Allocation
- Sentiment Analysis

Q: What is word sense disambiguation?

A: In computational linguistics, word-sense disambiguation (WSD) is an open problem of natural language processing, which governs the process of identifying which sense of a word (i.e. meaning) is used in a sentence, when the word has multiple meanings. The solution to this problem impacts other computer-related writing, such as discourse, improving relevance of search engines, anaphora resolution, coherence, inference *et cetera*.

Research has progressed steadily to the point where WSD systems achieve sufficiently high levels of accuracy on a variety of word types and ambiguities. A rich variety of techniques have been researched, from dictionary-based methods that use the knowledge encoded in lexical resources, to supervised machine learning methods in which a classifier is trained for each distinct word on a corpus of manually sense-annotated examples, to completely unsupervised methods that cluster occurrences of words, thereby inducing word senses. Among these, supervised learning approaches have been the most successful algorithms to date.

Q: What are topic models?

Q: What are topic models?

A: In machine learning and natural language processing, a topic model is a type of statistical model for discovering the abstract "topics" that occur in a collection of documents. Intuitively, given that a document is about a particular topic, one would expect particular words to appear in the document more or less frequently: "dog" and "bone" will appear more often in documents about dogs, "cat" and "meow" will appear in documents about cats, and "the" and "is" will appear equally in both. A document typically concerns multiple topics in different proportions; thus, in a document that is 10% about cats and 90% about dogs, there would probably be about 9 times more dog words than cat words. *A topic model captures this intuition in a mathematical framework, which allows examining a set of documents and discovering, based on the statistics of the words in each, what the topics might be and what each document's balance of topics is*.

Although topic models were first described and implemented in the context of natural language processing, they have applications in other fields such as bioinformatics.

Q: What are parts of speech?

Q: What are parts of speech?

A: In grammar, a part of speech (also a word class, a lexical class, or a lexical category) is a linguistic category of words (or more precisely *lexical items*), which is generally defined by the syntactic or morphological behaviour of the lexical item in question. Common linguistic categories include *noun* and *verb*, among others. There are open word classes, which constantly acquire new members, and closed word classes, which acquire new members infrequently if at all.

Q: What are parts of speech?

A: In grammar, a part of speech (also a word class, a lexical class, or a lexical category) is a linguistic category of words (or more precisely *lexical items*), which is generally defined by the syntactic or morphological behaviour of the lexical item in question. Common linguistic categories include *noun* and *verb*, among others. There are open word classes, which constantly acquire new members, and closed word classes, which acquire new members infrequently if at all.

Almost all languages have the lexical categories noun and verb, but beyond these there are significant variations in different languages. For example, Japanese has as many as three classes of adjectives where English has one; Chinese, Korean and Japanese have nominal classifiers whereas European languages do not; many languages do not have a distinction between adjectives and adverbs, adjectives and verbs (see stative verbs) or adjectives and nouns, etc. This variation in the number of categories and their identifying properties entails that analysis be done for each individual language. Nevertheless the labels for each category are assigned on the basis of universal criteria.

Q: What is the Flesch-Kincaid index?

Q: What is the Flesch-Kincaid test?

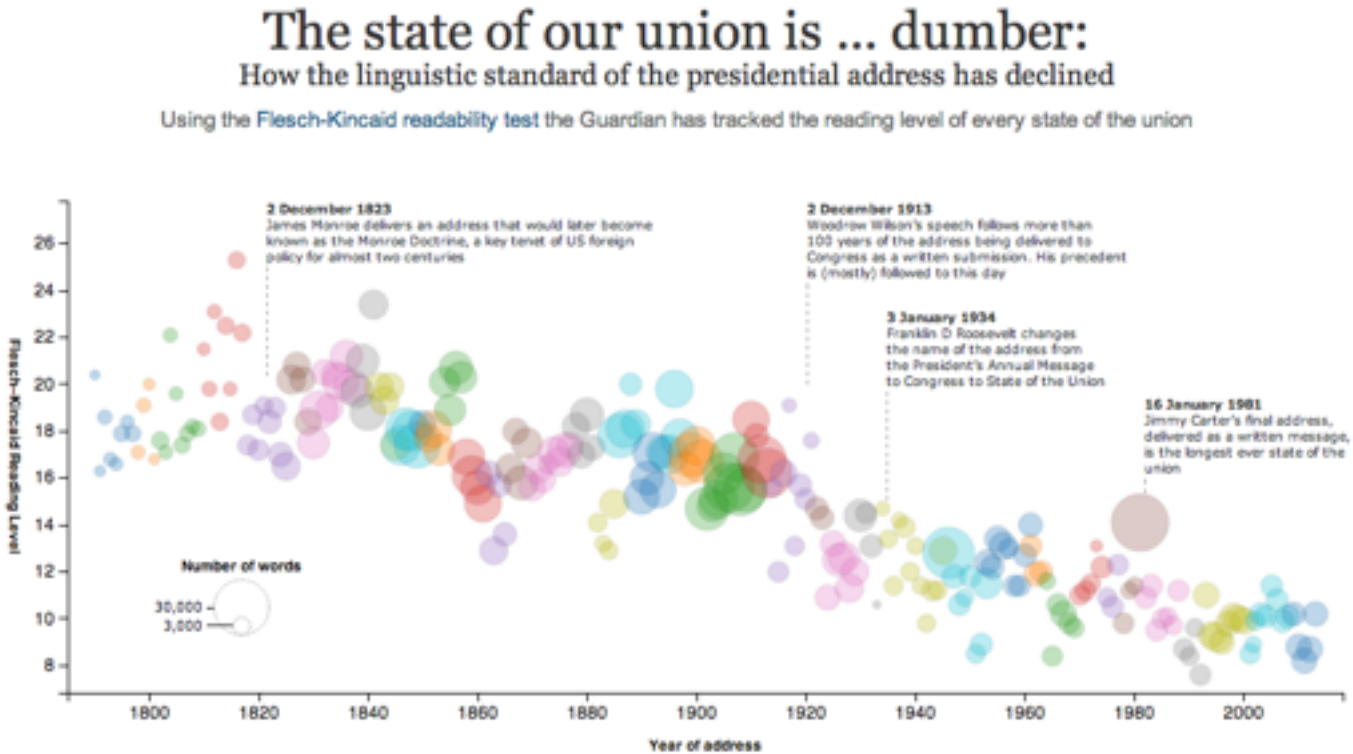
A: The Flesch/Flesch–Kincaid readability tests are readability tests designed to indicate comprehension difficulty when reading a passage of contemporary academic English. There are two tests, the Flesch Reading Ease, and the Flesch–Kincaid Grade Level. Although they use the same core measures (word length and sentence length), they have different weighting factors. The results of the two tests correlate approximately inversely: a text with a comparatively high score on the Reading Ease test should have a lower score on the Grade Level test. Rudolf Flesch devised both systems while J. Peter Kincaid developed the latter for the United States Navy. Such readability tests suggest that many Wikipedia articles may be "too sophisticated" for their readers.

In the Flesch Reading Ease test, higher scores indicate material that is easier to read; lower numbers mark passages that are more difficult to read. The formula for the Flesch Reading Ease Score (FRES) test is

$$206.835 - 1.015\left(\frac{\text{total words}}{\text{total sentences}}\right) - 84.6\left(\frac{\text{total syllables}}{\text{total words}}\right)$$ [8]

Scores can be interpreted as shown in the table below.

Score	Notes
90.0–100.0	easily understood by an average 11-year-old student
60.0–70.0	easily understood by 13- to 15-year-old students
0.0–30.0	best understood by university graduates


The state of our union is ... dumber:
How the linguistic standard of the presidential address has declined
Using the Flesch-Kincaid readability test the Guardian has tracked the reading level of every state of the union
2 December 1823
James Monroe delivers an address that would later become known as the Monroe Doctrine, a key tenet of US foreign policy for almost two centuries
2 December 1913
Woodrow Wilson's speech follows more than 100 years of the address being delivered to Congress as a written submission. His precedent is (mostly) followed to this day
3 January 1934
Franklin D Roosevelt changes the name of the address from the President's Annual Message to Congress to State of the Union
16 January 1981
Jimmy Carter's final address, delivered as a written message, is the longest ever state of the union
Number of words
30,000
3,000
Flesch-Kincaid Reading Level
26
24
22
20
18
16
14
12
10
8
1800
1820
1840
1860
1880
1900
1920
1940
1960
1980
2000
Year of address

Q: Latent Dirichlet Allocation (LDA) ?

Q: Latent Dirichlet Allocation (LDA) ?

A: In natural language processing, Latent Dirichlet Allocation (LDA) is a generative model that allows sets of observations to be explained by unobserved groups that explain why some parts of the data are similar. For example, if observations are words collected into documents, it posits that each document is a mixture of a small number of topics and that each word's creation is attributable to one of the document's topics. LDA is an example of a topic model and was first presented as a graphical model for topic discovery by David Blei, Andrew Ng, and Michael Jordan in 2003.

Q: Sentiment Analysis?

Q: Sentiment Analysis?

A: Sentiment analysis (also known as opinion mining) refers to the use of natural language processing, text analysis and computational linguistics to identify and extract subjective information in source materials.

Generally speaking, sentiment analysis aims to determine the attitude of a speaker or a writer with respect to some topic or the overall contextual polarity of a document. The attitude may be his or her judgment or evaluation (see appraisal theory), affective state (that is to say, the emotional state of the author when writing), or the intended emotional communication (that is to say, the emotional effect the author wishes to have on the reader).