CSC461
INTRODUCTION TO DATA SCIENCE

Dr. Muhammad Sharjeel

https://muhmmadsharjeel.github.io/

09

NATURAL LANGUAGE PROCESSING

NATURAL LANGUAGE PROCESSING AND DATA SCIENCE

- A large amount of data in real-world comes in the form of free text (unstructured)
  - Social media posts
  - Amazon reviews
  - Websites dumps
- Is it possible to use data science methods to get some meaningful information from unstructured text
  - Need to get some level of understanding what the text says
- Natural language processing could be used to write computer programs that can understand natural language

- Natural Language Processing (NLP) for data science primarily concerned with dealing with textual data
  - It is the intersection of linguistics, artificial intelligence, and computer science

UNDERSTANDING NATURAL LANGUAGE

- Understanding natural language is not an easy task for computers
- “I shot an elephant in my pajamas.” “How he got into my pajamas, I'll never know.”
- The Winograd schema challenge:
- “The city councilmen refused the demonstrators a permit because they [feared/advocated] violence.”
- Levesque argues that understanding such sentences requires more than NLP, but also commonsense reasoning and deep contextual reasoning

- But is it always hard?
- Two reviews for a movie
  - “Truly, a **stunning** exercise in large-scale filmmaking, a beautiful picture with **magnificent** and **marvelous** storytelling.”
  - “It's loud and full of vim, but a little **hollow** and **heartless**.”
- Which one is positive?
- Often very easy to tell the “overall gist” of natural language text

UNDERSTANDING NATURAL LANGUAGE

- Most of the time, the text has some signal (e.g., a sentiment)
  - Sometimes, it is easy to get that signal without truly understanding the text
- What type (domain) of texts are these?
  - Ronaldo replacement Ramos scores stunner against Switzerland
  - Leopard cubs reunited with mum
  - The new iPhone, from battery to always-on screen

SOME TERMINOLOGIES

- Documents: groups of free text
- Corpus: a collection of documents
- Terms: individual words
  - Separated by whitespace or punctuation
- Vocabulary: set of possible (unique) words in a language
- Syntax: Grammatical structure of language
  - Rules which forms sentences/expressions
- Semantics: Study of the meaning of language
- Example: John is rectangular and a rainbow.
  - Syntactically correct
  - Semantically meaningless

NLP TASKS

- Tokenization
  - Splitting text into tokens
- Lemmatization/Stemming
  - Stemming returns the stem of the word
  - Lemmatization returns the context in which the word is being used
- Morphological segmentation
  - Breaks words into morphemes
  - Easy for English, but other languages are difficult
- Part-of-Speech (PoS) tagging
  - Determining whether a word is a noun/adverb/verb etc.
- Parsing
  - A tree representation of different syntactic categories of a sentence
- Named Entity Recognition
  - Identifying key entities in a text
- Sentiment analysis
  - Deciding whether reviews/opinions are positive or negative

NLP TASKS

- Machine Translation
  - Translating from one language to another
- Text summarization
  - Condensing a piece of text to a shorter version
- Natural Language Generation
  - To produce natural language text
- Paraphrase identification
  - Identifying alternative linguistic expressions of the same meaning at different textual levels
- Text classification
  - Categorize text documents into one or more classes
- Author identification
  - Identifying the feasible author of unknown documents

REPRESENTING A DOCUMENT - IN MATH

- Many data science problems (tasks) do not require to truly understand the text
- Bag of Words and TFIDF
- Simple and useful methods to infer some meaning from the text without deep understanding

- In BoW, the documents are represented as vectors of word frequencies
- It is simply an unordered collection of words and their frequencies (counts)
- Order of words does not matter, just occurrences

REPRESENTING A DOCUMENT - IN MATH

- Example: Following are three sample reviews about a movie:
  - Review 1: This movie is very scary and long
  - Review 2: This movie is not scary and is slow
  - Review 3: This movie is spooky and good
- Vocabulary (all the unique words) in the three reviews
  - 11 unique words: ‘This’, ‘movie’, ‘is’, ‘very’, ‘scary’, ‘and’, ‘long’, ‘not’, ‘slow’, ‘spooky’, ‘good’
- For BoW, take each of vocabulary word and mark its occurrence in the 3 movie reviews

	This	movie	is	very	scary	and	long	not	slow	spooky	good	Total length
R1	1	1	1	1	1	1	1	0	0	0	0	7
R2	1	1	2	0	0	1	1	0	1	0	0	8
R3	1	1	1	0	0	0	1	0	0	1	1	6

- Vector R1: [1 1 1 1 1 1 1 0 0 0 0]
- Vector R2: [1 1 2 0 0 1 1 0 1 0 0]
- Vector R3: [1 1 1 0 0 0 1 0 0 1 1]

REPRESENTING A DOCUMENT - IN MATH

- Drawbacks of BoW
  - When data (text) is added containing new words, vocabulary size would increase and the length of the vectors would also increase
  - Vectors would have many 0s, resulting in a sparse matrix
  - No information on the grammar of the sentences nor on the ordering of the words in the text

REPRESENTING A DOCUMENT - IN MATH

- Term frequency refers to the counts of each word in a document
- Denoted $tf_{i,d}$ = frequency of (term) word *i* in document *d*
- $tf_{i,d}$ = number of times word (term) *i* appears in a document *d* / total number of words (terms) in document *d*
- Often the raw count, but there are also other possibilities

$tf_{i,d} \in \{0,1\}$ – does word occur in document or not

$\log (1 + tf_{i,d})$ – log scaling of counts

$tf_{i,d} / max_i * tf_{i,d}$ – scale by document's most frequent word

REPRESENTING A DOCUMENT - IN MATH

- Example: R2: This movie is not scary and is slow
  - Vocabulary: ‘This’, ‘movie’, ‘is’, ‘very’, ‘scary’, ‘and’, ‘long’, ‘not’, ‘slow’, ‘spooky’, ‘good’
  - Total number of words (terms) in R2 = 8
  - tf for term ‘This’ = (number of times ‘this’ appears in R2) / (total number of terms in R2) = 1/8
  - tf(‘movie’) = 1/8
  - tf(‘is’) = 2/8 = 1/4
  - tf(‘very’) = 0/8 = 0
  - tf(‘scary’) = 1/8
  - tf(‘and’) = 1/8
  - tf(‘long’) = 0/8 = 0
  - tf(‘not’) = 1/8
  - tf(‘slow’) = 1/8
  - tf(‘spooky’) = 0/8 = 0
  - tf(‘good’) = 0/8 = 0

REPRESENTING A DOCUMENT - IN MATH

- Term frequencies for all the words (terms) in all the 3 reviews

	This	movie	is	very	scary	and	long	not	slow	spooky	good
tf - R1	1/7	1/7	1/7	1/7	1/7	1/7	1/7	0	0	0	0
tf - R2	1/8	1/8	1/4	0	1/8	1/8	0	1/8	1/8	0	0
tf - R3	1/6	1/6	1/6	0	0	1/6	0	0	0	1/6	1/6

REPRESENTING A DOCUMENT - IN MATH

- Term frequencies tend to be “overloaded” with very common words (e.g., “the”, “is”, “of”, etc)
- Inverse document frequency weight words negatively in proportion to how often they occur in the entire set of documents
- idf_i = log (total number of documents / number of documents with word (term) *i*)
- IMPORTANT: Inverse document frequency is just defined for terms not for term–document pairs
- As with term frequency, there are other version as well with different scaling, but the log scaling is the most common

REPRESENTING A DOCUMENT - IN MATH

- Example: R2: This movie is not scary and is slow
  - idf for term ‘This’ = log(total number of documents) / (number of documents containing the term ‘This’)
    = log(3/3) = log(1) = 0
  - idf(‘movie’) = log(3/3) = 0
  - idf(‘is’) = log(3/3) = 0
  - idf(‘not’) = log(3/1) = log(3) = 0.48
  - idf(‘scary’) = log(3/2) = 0.18
  - idf(‘and’) = log(3/3) = 0
  - idf(‘slow’) = log(3/1) = 0.48

- Inverse document frequencies for all the words (terms)

	idf
This	0
movie	0
is	0
very	0.48
scary	0.18
and	0
long	0.48
not	0.48
slow	0.48
spooky	0.48
good	0.48

REPRESENTING A DOCUMENT - IN MATH

- Term frequency inverse document frequency = $tf_{i,d} \times idf_i$
- Simply a product of tf and idf scores for each word in the corpus (dataset)
- Values are higher when both TF and IDF values are high, i.e., the word is rare in all the documents combined but frequent in a single document
- Words (terms) with a higher score are more important
  - Also common to remove “stop words” beforehand
  - Seems to work much better than using raw scores, e.g., computing similarity between documents or building machine learning classifiers on the documents

REPRESENTING A DOCUMENT - IN MATH

- Example: R2: This movie is not scary and is slow
  - tf-idf for word (term) ‘This’ = tf(‘This’) * idf(‘This’) = 1/8 * 0 = 0
  - tf-idf(‘movie’) = 1/8 * 0 = 0
  - tf-idf(‘is’) = 1/4 * 0 = 0
  - tTf-idf(‘not’) = 1/8 * 0.48 = 0.06
  - tf-idf(‘scary’) = 1/8 * 0.18 = 0.023
  - tf-idf(‘and’) = 1/8 * 0 = 0
  - tf-idf(‘slow’) = 1/8 * 0.48 = 0.06

REPRESENTING A DOCUMENT - IN MATH

- Term frequency inverse document frequencies for all the words (terms) in all the 3 reviews

	tf*idf(R1)	tf*idf(R2)	tf*idf(R3)
This	0	0	0
movie	0	0	0
is	0	0	0
very	0.068	0	0
scary	0.025	0.022	0
and	0	0	0
long	0.068	0	0
not	0	0.060	0
slow	0	0.060	0
spooky	0	0	0.080
good	0	0	0.080

REPRESENTING A DOCUMENT - IN MATH

- BoW just creates a set of vectors containing the count of word occurrences in the document (reviews)
- TF-IDF contains information on the more important words and the less important ones

- Bag of Words vectors are easy to interpret
- TF-IDF usually performs better in machine learning models

- Both give no understanding of the contextual information of words
  - For example, when detecting similarity between the words ‘spooky’ and ‘scary’, or translating text into another language, requires contextual information

REPRESENTING A DOCUMENT - IN MATH

- Try it yourself, for the following three sentences, calculate tf*idf for each word (term)
  - S1: The goal of this lecture is to explain the basics of text processing
  - S2: The bag of words model is one such approach
  - S3: In this lecture we learned text processing via bag of words

MEASURING SIMILARITY

- Given two vectors v and w, need to measure their similarity
- Mostly measures are based on dot product or inner product (linear algebra)
  - dot-product $(\vec{v}, \vec{w}) = \vec{v} \cdot \vec{w} = \sum_{i=1}^{N} \vec{v}\vec{w}$
- High (1): when two vectors have values in the same dimensions
- Low (0): when two vectors are orthogonal

- Do-product is longer if vector is longer
  - vector length = $|\vec{v}| = \sqrt{\sum_{i=1}^{N} v_i^2}$
- Vectors are longer if they have higher value in each dimension

MEASURING SIMILARITY

- To normalize, divide the dot-product by the length of the two vectors

$$\frac{\vec{v} \cdot \vec{w}}{|\vec{v}|\,|\vec{w}|}$$

- It is the same as the cosine of the angle between them

$$\vec{v} \cdot \vec{w} = |\vec{v}|\,|\vec{w}| \cos\theta$$

$$\frac{\vec{v} \cdot \vec{w}}{|\vec{v}|\,|\vec{w}|} = \cos\theta$$

- Given two text documents x and y, represented by their TF-IDF vectors (or any vectors), the cosine similarity is
- Cosine similarity(x, y) = $\frac{x^T y}{|x| \times |y|}$
- Formally, it measures the cosine of the angle between two vectors x and y
- $\cos(0^o) = 1$, $\cos(90^o) = 0$
- Similar documents have high cosine similarity, dissimilar documents have low

MEASURING SIMILARITY

- Example:
- cos(text1, text2) = (text1 . text2) / |text1| |text2|

- text1 = [0, 3, 0, 0, 2, 0, 0, 2, 0, 5]
- text2 = [1, 2, 0, 0, 1, 1, 0, 1, 0, 3]

- text1 . text2 = 0*1 + 3*2 + 0*0 + 0*0 + 2*1 + 0*1 + 0*0 + 2*1 + 0*0 + 5*3 = 25

- |text1| = (0*0 + 3*3 + 0*0 + 0*0 + 2*2 + 0*0 + 0*0 + 2*2 + 0*0 + 5*5) ^ 0.5
  = (42) ^ 0.5 = 6.481
- |text2| = (1*1 + 2*2 + 0*0 + 0*0 + 1*1 + 1*1 + 0*0 + 1*1 + 0*0 + 3*3) ^ 0.5
- = (17) ^ 0.5 = 4.12

- cos(text1, text2) = 0.94

LANGUAGE MODELS

- While the BoW model is surprisingly effective, it is clearly throwing away a lot of information about the text
- Example: “boring movie and not great” is not the same as “great movie and not boring”
  - However, they have the exact same BoW representations
- Language models try to build a more accurate model of how words really relate to each other
- A (probabilistic) language model aims at providing a probability distribution over every word, given all the words before it

  P (word$_i$| word1, … , word$_{i-1}$)
- Example: Can you guess what the next word should be
  - Data science is the study and practice of how we can extract insight and knowledge from large amounts of

LANGUAGE MODELS

- N-gram language model: the probability of a word depends only on the n − 1 words preceding it
- P (word$_i$ | word1, … , word$_{i-1}$) ≈ P (word$_i$ | word$_{i-n+1}$, … , word$_{i-1}$)
- Given a sequence of n-1 words, an n-gram language model tries to predicts the most probable word that might follow this sequence
  - Useful in many NLP applications including speech recognition, machine translation, predictive text input etc.
- Example: "There was heavy rain" vs. "There was heavy flood”
- From experience (previous knowledge), the former sentence sounds better
- An n-gram language model will try to learn this (the probability) from the training corpus (previous knowledge)

- This puts a hard limit on the context that we can use to make a prediction, but also makes the modeling more tractable
  - “large amounts of data” vs. “large amounts of hotdogs”

LANGUAGE MODELS

- If the model simply relies on how often a word occurs without looking at previous words, it is called unigram
- If the model considers only the previous word to predict the current word, then it's called bigram
- If two previous words are considered, then it's a trigram

- Example: This is a data science course taught to BCS students
- Unigrams: This, is, a, data, science, course, taught, to, BCS, students
- Bigrams: This is, is a, a data, data science, science course, course taught, taught to, to BCS, BCS students
- Trigrams: This is a, is a data, a data science, data science course, science course taught, course taught to, taught to BCS, to BCS students

LANGUAGE MODELS

- P (the | its water is so transparent that)
- The probability of word ‘the’, given ‘its water is so transparent that’
- To estimate the probability function is through the relative frequency count approach
  - Take a substantially large corpus, count the number of times ‘its water is so transparent that’ occurs, and then count the number of times it is followed by ‘the’
- Formally
  - P (the | its water is so transparent that) = C (its water is so transparent that the) / C (its water is so transparent that)

THANKS