INTRO TO NLP

An Introduction to Natural Language Processing.

INTRODUCTION

Objective: use speech and words along with computer run algorithms.

Sentiment Analysis - How positive or negative is text about a topic?

Prediction - What genres should Netflix classify a movie as to maximize views? Based on product reviews, can we predict the star rating of a product?

Translation - Recognize words in one language to provide similar words in another.

Playground: https://www.deepl.com/translator

Summarization - Take a long document and produce a shorter one (a synthesis) without losing meaningful information.

Methods: The main idea is to quantify the occurrence of relevant words and, based on the context, to map them into vectors. That is to say that we want to create mathematically representable quantities from words and text; they will serve as features for data analysis. One approach is separate the text data into sentences and then sentences can be used to extract (key) words and expressions.

REGULAR EXPRESSIONS

Goal: provide a framework that allows us to search for different text strings.

For example, Regular Expressions (frequently called “regex”) allows us to label all tweets with a “1” if they contain the following list of words:

- college
- College of
- colleges
- The College

The idea is to detect that in all expressions above we have the same concept "college".

Examples of common REGEX patterns

[tT]imber - would match lower or uppercase T

[A-Z] - would match any capital character

[a-z] - would match any lowercase character

[0-9] - would match any single number (i.e., 9)

[^A-Z] - would match anything that isn’t an uppercase letter.

\w - would match any letter.

A comprehensive manual on regex can be found here:
https://www3.ntu.edu.sg/home/ehchua/programming/howto/Regexe.html

EXAMPLE

A sample post from the *old* Twitter:

Discussion: If you were to apply regular expressions and association rules, how would you establish the *sentiment* of this tweet?

Critical thinking: What are the keywords in this communication?

THE BAG OF WORDS MODEL (BOW)

Main Goal: use concurrences within context and counts of keywords to make predictions.

Observation: there are many words that do not matter (such as prepositions or definite and indefinite articles).

Important: each word can be translated into a binary value of occurrence.

Analog Example:

Statement 1: Jurassic World was the pinnacle of human achievement.

Statement 2: Human kind would be better without Jurassic World.

Statement 1: Jurassic World was the pinnacle of human achievement.
Statement 2: Human kind would be better without Jurassic World.

Discussion: How can you tell that the different statements reflect different sentiments?

THE BAG OF WORDS MODEL (BOW)

	Human Kind Better Without Jurassic World.	Jurassic World Pinnacle Human Achievement.
Achievement	0	1
Better	1	0
Human	1	1
Jurassic	1	1
Kind	1	0
Pinnacle	0	1
Without	1	0
World	1	1
Sentiment	**Negative**	**Positive**

Issues:

- If some sentences are much longer in length, the vocabulary would increase and as such, the length of the vectors would increase; this is a dimensionality problem.
- The new sentences may contain more different words from the previous sentences.
- The vectors would also contain many zeros, thereby resulting in a sparse matrix.
- No information on the grammatical structure or the actual ordering of the words is being used.

TERM FREQUENCY-INVERSE DOCUMENT FREQUENCY

The term frequency-inverse document frequency is a measure that quantifies the importance of a word in the context of a document or a *corpus*.

The *term-frequency* of a word is the relative frequency of the term in the context of the document.

$$\mathrm{TF}(t,d) := \frac{\#\text{ of times the term appears in the document}}{\#\text{ of terms in the document}}$$

The *inverse document frequency* is defined as:

$$\mathrm{IDF}(t,d) := \log\left(\frac{\#\text{ of documents}}{\#\text{ of documents with term } t}\right)$$

The quantification of relative importance is defined as the product of TF and IDF.

TF-IDF gives larger values for less frequent words and is high when both IDF and TF values are high, for instance the word is rare in all the documents combined but frequent in a single document.

Python example.

PRE-PROCESSING

We assume that, up to this point, we imported the data (both reviews and corresponding start ratings.) The reviews are stored in a variable that has the same name.

We are considering a step process:

- change all characters to lower case
- remove punctuation
- separate the different words in each review
- discard the "stopwords"
- lemmatize or stem the remaining words.

In the context of **natural language processing (NLP)**, a **token** is a basic unit of text used in computational analysis. The process of breaking down text into these units is called **tokenization**.

TOKENS

Definition: Depending on the tokenization strategy used, a token can represent a word, subword, or character. Tokens are the building blocks for further processing in NLP tasks.

Types of Tokens:

- **Word Tokens**: Each word in a sentence is treated as a token.
  Example: `"I love NLP."` → `["I", "love", "NLP", "."]`
- **Subword Tokens**: Words are broken into smaller meaningful units, often used in models like BERT or GPT to handle rare or out-of-vocabulary words.
  Example: `"unbelievable"` → `["un", "##believ", "##able"]`
- **Character Tokens**: Each character is treated as a token.
  Example: `"cat"` → `["c", "a", "t"]`

TOKENS

Purpose:

Tokens enable the conversion of human-readable text into numerical representations that can be processed by machine learning algorithms.

Usage in Models:

- Most modern NLP models like **transformers** use tokenized text as input.
- Special tokens are often added for specific purposes, such as:
  - `<CLS>`: Represents the start of a sequence (e.g., for classification tasks).
  - `<SEP>`: Separates sequences (e.g., in question-answering tasks).
  - `<PAD>`: Used for padding sequences to ensure uniform input lengths.

Challenges:

- Handling multi-word expressions or compound words.
- Managing languages with no spaces (e.g., Chinese, Japanese).
- Reducing the impact of tokenization errors on downstream tasks.

PROBABILISTIC LANGUAGE MODELING

IMPORTANT The conditional probability rule:

$$P(A \cap B) = P(A) \cdot P(B|A)$$

Goal: Evaluate the probability that a sequence of words such as $(w_1, w_2, w_3, ...w_n)$ occurs:

$$\mathrm{P}(\text{Sentence}) = \mathrm{P}(w_1, w_2, w_3, ...w_n) = \mathrm{P}(w_1) \cdot \mathrm{P}(w_2, w_3, w_4 ...w_n | w_1) = \ldots$$

Smartphones use this information to predict what the next word you will type will be, for example:

$$\begin{aligned}
\mathrm{P}(w_1, w_2, w_3, w_4) &= \mathrm{P}(w_1) \cdot \mathrm{P}(w_4, w_3, w_2 | w_1) \\
&= \mathrm{P}(w_1) \cdot \mathrm{P}(w_2 | w_1) \cdot \mathrm{P}(w_4, w_3 | w_2, w_1) \\
&= \mathrm{P}(w_1) \cdot \mathrm{P}(w_2 | w_1) \cdot \mathrm{P}(w_3 | w_1, w_2) \cdot \mathrm{P}(w_4 | w_1, w_2, w_3)
\end{aligned}$$

which mean the probability of word w_4 provided the words w_1, w_2 and w_3 occurred.

Critical thinking: How do we compute these probability values?

PROBABILISTIC LANGUAGE MODELS

Example: P(today | It, is, sunny) = 50%

The model you use to predict is called the “language model”

Important Concept: The Conditional Probability Rule states that probabilities of an events in the future are defined by the multiplication of all (conditional) probabilities leading to that given event.

P(Today, it, was, sunny) = P(Today) P(it | Today) P(was | Today, it) P(sunny | Today, it, was)

P(Today, is, the, fiftennth) = P(Today) P(is | Today) P(the | Today, is) P(fifteenth | Today, is, the)

- **Unigram Models**: P(rainy | Today, it, was) ~ P(Today) P(it) P(was)
- **Bigram Models**: P(rainy| Today, it, was) ~ P(rainy | was)
- **N-gram models**: Same as the above, but for arbitrary distances. For example a tri-gram: P(rainy | Today, it, was)

Often used in nested ways (i.e., a 3-gram model + unigram).

EVALUATIVE METRICS - PERPLEXITY

One might expect a model to be good at predicting cold in this sentence:

"It is cold."

And not as good at predicting:

"It is very cool outside when the winter is cold"

For a variety of reasons; the biggest is the complexity/length of the sentence.

• Perplexity is a measurement of how well a probability model predicts a test data. In the context of Natural Language Processing, perplexity is one way to evaluate language models.

• Low perplexity is good, and high perplexity is bad since perplexity is the exponentiation of an **entropy**.

• The goal is to minimize Perplexity(W).

Calculation of perplexity for a full a sequence of words:

$$\sqrt[n]{\prod_{i=1}^{n} \frac{1}{P(w_i|w_1 w_2 \ldots w_{i-1})}}$$

WORD EMBEDDINGS - GLOVE

Reference: https://nlp.stanford.edu/projects/glove/

Example for using the vector words: monarch - man = queen.

The main idea is that we can do more than just counting occurences but rather represent the words from the vocabulary of a language as vectors whose entries are real numbers. As such, the GloVe algorithm is analysing word co-occurrencies within a text corpus; the steps are as follows:
A *co-occurence* matrix X is created where its entries X_{ij} represent how often word i is present in the context of the word j. Thus there is a parsing of the corpus for building the matrix X and then the model is constructed based on this matrix.

For the words i and j we create vectors $\vec{w}_i$ and $\vec{w}_j$ such that

$$\vec{w}_i^T \cdot \vec{w}_j + b_i + b_j = \log(Xij)$$

where b_i and b_j are bias terms (i.e. intercept terms for a regression model). We want to build word vectors that retain useful information of how words i and j co-occur.

PROBABILISTIC LANGUAGE MODELS

In order to determine the entries for the $\vec{w}_i$, *we minimize the following objective function*

$$J := \sum_{i=1}^{V} \sum_{j=1}^{V} f(X_{ij}) \left(\vec{w}_i^T \cdot \vec{w}_j + b_i + b_j - \log(Xij) \right)^2$$

The function f is chosen in order to prevent the skewing of the objective function by the words that co-occur too often. In this sense a choice for the function f could be

$$f(X_{ij}) := \begin{cases} \left(\frac{X_{ij}}{x_{max}} \right)^{\alpha} \text{ if } \ X_{ij} < x_{max} \\ 1 \quad \text{otherwise} \end{cases}$$

where α and x_{max} can be adjusted by the user.

APPLICATIONS

- Sentiment Analysis
- Speech Recognition
- **Information** Retrieval
- Question Answering