Machine Learning Applied to Natural Language Processing

May 20, 2020. Catarina M. Rodrigues
Barreiro School of Technology/ESTBarreiro, IPS.
Prof. Jacinto Estima
Prof. Vala Rohani

Introduction

Natural Language Processing (NLP) is a multidisciplinary field combining linguistics, artificial intelligence and computer science. Its main goal is the processing and interpreting natural language automatically – this includes interpreting sentiment, knowing which parts and words are relevant, comparing writing styles, and ultimately understanding all the nuances that compose human communication.

Knowledge of language and discourse conventions is needed to understand ambiguities and engage in complex language behavior. This is why speech and text recognition computational techniques must encompass phonetics and phonology (the study of linguistic sounds), morphology (meaningful components of words), syntax (structural relationship between words), semantics (meaning), pragmatics (how language is used to accomplish goals) and discourse (linguistic units larger than a single utterance).

Most of the tasks in speech and language processing can be viewed as methods to resolve the ambiguity present in language. The challenges in Natural Language Processing involve two processes or subsets:

- Natural Language Understanding – understands the meaning of given text, including the nature and structure of each word inside text. NPL tasks should resolve these problem of ambiguity:
    - ✔ Words can have multiple meanings (Lexical)
    - ✔ Sentences can have multiple meanings (Semantic)
    - ✔ Sentences can have multiple parse trees (Syntactic). Parse trees represent the syntactic structure of a string according to some context-free grammar.
    - ✔ A previously mentioned phrase or word has a different meaning (Anaphoric)

- Natural Language Generation – the process of generating text from structured data, with meaningful phrases and sentences.
    - ✔ Text Planning – planning and ordering of structured data.
    - ✔ Sentence Planning – sentences and words are combined to represent the flow of information.
    - ✔ Realization – production of grammatically correct sentences to convey meaning and represent text.

NLP offers many practical applications, as the modern real world revolves around communication and human-computer interactions. The ability to recognize, interpret meaning and generating a relevant response can be useful in some of the areas below.

Healthcare

From historical records and patient's own speech, NLP can aid in diagnosis and treatment of patients, by recognizing and predicting diseases, providing a more efficient approach to healthcare for the general population. Chatbots therapists are also being used in serving patients with anxiety, depression or other mental disorders, providing convenient and impersonal opportunities to those who most need it.

Business and Marketing

By applying sentiment analysis to social media, interviews, reviews, and customer surveys we can extract relevant information about the consumers' choices and what drives their decisions, what attracts and what repels them – this is fundamental in offering competitiveness to organizations through applying the learned information to marketing strategies. NLP methods can also be applied to the human resources field, by automating the recruitment of potential hires.

Identifying Spam and Fake News

Some companies (eg. Google or Yahoo) already have NLP methods (eg. tokenizing spam terms and words) integrated in their servers to analyze text in emails, filter them as spam or not spam before they even reach the recipient's inbox. Following the same train of thought, a similar approach can be used in identifying fake, disingenuous or biased news.

Chat bots and other Text and Voice-driven interfaces

Artificial intelligence bots and voice interfaces (eg. Cortana, Alexa, Siri…) currently apply NLP methods to serve their customers on the most menial tasks – by responding to vocal prompts, the interface over time creates a personalized repository of speech information about a customer, plus with internet access, they can offer assistance in day-to-day tasks, such as playing a favorite song, reminding the customer of activities or facts, telling the weather forecast, report news, make reservations, etc.

A general NLP pipeline consists of the following:

1. Text extraction
2. Data Pre-processing
3. Machine Learning algorithms
4. Interpretation of results

This paper will review and synthesize the most covered strategies in available literature to solve Natural Language Processing problems, presenting in a pragmatic way the general pipeline. The different NLP techniques and algorithms I will review and explore in this study are the following:

- Tokenization
- Bag of Words
- Stemming
- Lemmatization
- Sentiment Analysis
- Topic Modeling
- Text Generating

Literature Review

1. Data Pre-Processing

Text pre-processing encompasses selecting, and cleaning and transforming the text data to solve our problem. For this effect, we can apply syntactic analysis to decide the starting and ending of each sentence, so that any string of characters from the start and up to the full stop symbol is taken as one full sentence.

Semantic analysis are used to understand the role of every word in a sentence, such as nouns, verb, adjective, adverb, etc. This process is called Part-Of-Speech tagging.

Some of the most widely used techniques of text pre-processing are further described below, ordered as a typical NLP pipeline. The collection of documents used as data set is called a corpus, and the collection of words or sequences we select as relevant in our index is called a vocabulary or lexicon.

1.1 Tokenization

Analyzing the words in text data is one of the first and most basic steps of the overall NLP strategy. Tokenization is a particular kind of segmentation – the splitting of phrases or paragraphs into smaller, individual segments. The segments can be a document divided into paragraphs, a paragraph divided into sentences, sentences into phrases, or phrases into words (tokens). After tokenization we can count the number of words and its frequency in a text, for further organization and classification in levels of importance. This ultimately turns an unstructured string into a numerical data structure, best suited for machine learning.

Methods of tokenization:

- Python’s split() function method, using whitespace as a “delimiter”
- re.compile() and re.findall() with Regular Expressions library (RegEx) in Python
- word_tokenize() with Natural Language Toolkit library (NLTK) in Python
- create_pipe() with open-source spaCy library for advanced NLP
- text_to_word_sequence() with open-source Keras library for deep learning
- tokenize() and split_sentences() with open-source library Gensim for unsupervised topic modeling and NLP

1.1.2 N-grams

N-grams are sequences of N-words. Bigrams are two words frequently occurring together in a document (example: “Las Vegas”. Trigrams are three words frequently occurring, and so on (example: “The Three Musketeers”). We usually create specific N-gram models to decide which N-words can be chunked together to form single entities. This can help in word predictions and to make spelling error corrections.

An N-gram model basically predicts the occurrence of a word, based on the occurrence of its N-1 previous words. Gensim’s Phrases model can build and implement the bigrams, trigrams, quadgrams and more.

1.2 Bag of Words (BOW)

This is a NLP model used to represent text data in fixed-length vectors or matrix by counting how many times each word appears in the document. This model is only concerned if previously chosen known words (tokens) occur in the document.

Let's assume we ignore the order and grammar of the words, and put them all jumbled into a "bag", one bag for each sentence or document to summarize while capturing the essence of a document.

After we define our vocabulary by tokenization, we can score how many times the tokens occur in a document, and does not specify order or place.

The simplest scoring method is to mark the presence of words as a boolean value, 0 for absent, 1 for present. As the scale of the document text increases, so does the length of the vectors. We may come across lots of zero scores, as variability can make some documents contain very few tokens – this results in a sparse vector, with lots of zeros, which require more memory and computational resources to process.

It is considered best practice to decrease the size and complexity of the vocabulary when using a BOW model, by normalizing it. Some techniques to solve this problem include:

- Removal of stop words (commonly used unimportant words in extracting meaning from the overall text, such as "the", "a", "of", etc…). Stop words can also contain useful information, and discarding them is not always helpful.
- Ignoring upper/lower case, punctuation and numbers.
- Fixing misspelled words.
- **Stemming** – the reduction of inflected words to their root form (eg. "caring" becomes "car"). This is all about removing suffices from words in an attempt to combine words with similar meanings together under their common stem. A stem doesn't need to be a correctly spelled word, but merely a label or token representing different spellings of a word.
- **Lemmatization** – removal of inflection of words, taking the morphology into consideration (eg. "caring" becomes "care"). Because we need to normalize a word down to its semantic root – its lemma – it will require more computational power and language expertise than Stemming.

In the context of information retrieval (chatbots, search engines, etc), these vocabulary compression approaches can reduce the precision and accuracy as there will be more documents associated with the same query words, yet not relevant to the words' original meanings.

The BOW model is very simple to implement, although it is only as good as the defined vocabulary or tokens – the size and sparsity of vectors, and the discarding of word order are some of the shortcomings to have in mind when using this model. Given *m* documents and *n* words in the available vocabulary, we can create a document-term matrix.

Methods of implementing Bag of Words (BOW)

- Tokenizer() with open-source Keras library for deep learning
- Count Vectorizer with scikit-learn in Python (document-term matrix)

1.2.2 TF-IDF Vectorization

While Bag of Words are vectors of word counts or frequencies, TF-IDF vectors are word scores that better represent their importance, and count the exact spellings of terms in a document. These are both frequency-based statistical models that can be applied separately or as part of an NLP pipeline.

Merely counting words can't be as descriptive as using their TF-IDF score. TF-IDF matrices, or document-term matrices have been the mainstay of search and information retrieval for decades. Every web-scale search engine with millisecond response times has a TF-IDF document-term matrix hidden behind it.

2. Advanced Natural Language Processing Techniques

2.1. Sentiment Analysis

Sentiment Analysis is the process of assigning sentiment scores such as positive, negative or neutral, to the topics, themes and categories within a sentence. There are two approaches to sentiment analysis:

- A machine learning model that learns from data.

Relies on a labeled set of statements of documents to train a model to create the rules. We would need a lot of data, text labeled with the "right" sentiment score. An example of data used for this approach are #hashtags, often used to create a self-labeled dataset.

- A rule-based algorithm composed by a human.

Uses human-designed rules (heuristics) to measure sentiment. (There are already implemented available sentiment libraries scored by linguists, with respective polarity and subjectivity scores)

After cleaning and organizing our data, we must define the sentiment-bearing phrases or components and assign them a score (-1 to +1). These sentiment-bearing components will compose a sentiment library, it should have a large collection of adjectives and phrases that have been hand-picked and scored by humans; this can be a lengthy and tricky process as different people can attribute different scores to the same component. Multi-layered approaches enable sentiment scores to cancel each other out.

After defining the sentiment library, some rules or guidelines for evaluation of sentiment expressed towards a particular component can be set, based on its proximity to positive or negative words. No rule-set can account for every abbreviation, acronym or double-meaning that may appear in any given text document, so a purely rules-based system is something that should be avoided.

Usually, the resulting hit counts are resolved by log odds ratio operation, which returns a sentiment score for each phrase on a scale of 1 (very negative) to +1 (very positive).

Methods of implementing Sentiment Analysis

- TextBlob().sentiment with TextBlob library in python, built on top of NLTK

- VADER algorithm – Valence Aware Dictionary for Sentiment Reasoning (NLTK package has an implementation of this algorithm)

2.2. Topic Modeling

This is a method for uncovering topics and themes in text data, by clustering words and assigning them values based on their distribution. Let's review some of the most popular topic modeling techniques:

Latent Semantic Analysis (LSA)

When you use this tool, not only can you represent the meaning of words as vectors, but you can use them to represent the meaning of entire documents. This involves replacing counts in the generated document-term matrix with a TF-IDF score (term frequency times inverse document frequency. Term frequencies are the counts of each word in a document; Inverse frequency means that you'll divide each of the word counts by the number of documents in which the word occurs), which is used as a weighting factor. To reduce the matrix's sparsity, redundancy and noise, we perform dimensionality reduction with singular value decomposition technique. LSA is quick and efficient, but needs large sets of documents to get accurate results.

Probabilistic Latent Semantic Analysis (PLSA) uses a probabilistic method instead of singular value decomposition to solve for sparsity and noise.

Latent Dirichlet Allocation (LDA)

One of the most popular topic modeling techniques. LDA finds groups of related words by assigning each word to a random hidden topic, as many as the user chooses. It uses Dirichlet distributions, and assigns each document to a mixture of topics – this means each document can be described by probability distributions of topics. After assigning every word to a topic, the algorithm goes multiple times through each word iteratively and reassigns it to a topic taking into consideration how often the topic occurs in the document, and how often the word occurs in the topic overall. LDA algorithm require you to reprocess the entire corpus every time you add a new document, therefore being slower.

Methods of implementing topic modeling

- TfidfVectorizer, TruncatedSVD modules with scikit-learn
- LdaModel module with open-source library Gensim for unsupervised topic modeling and NLP

2.3. Text Generation

The goal is to generate text that is indistinguishable from human-written text. This can be achieved with Markov chains at its simplest form, or deep learning and Long Short-Term Memory (LSTM). In markov chains, the next state of a process only depends on the previous state, so the scope is very limited in generating text. LSTM takes into account not only the previous word, but other things such as the words before the previous word, what words have already been generated, etc.

Methods of implementing Text generation

- LSTM with open-source Keras library for deep learning

3. Available Tools for Natural Language Processing

- **Natural Language Toolkit** (NLTK) is one of the most popular suite of libraries for dealing with a variety of NLP tasks.
- **SpaCy** is a free open source library for advanced NLP tasks in Python. It is best suited for dealing with large volumes of text data. It is fast and easy to use and very well documented, both for research and teaching purposes.
- **TextBlob** is another Python library, intuitive and user friendly, and ideal for beginners
- **MonkeyLearn** is a SaaS platform for advanced NLP tasks, that lets you build customized models to solve problems.
- **Gensim** is a highly specialized open source library that is more directed at topic modelling and document similarity. It is best used in conjunction with other libraries.
- **Keras** is a library best used for text processing, such as tokenization, text classification and text generation.

4. Implementation Project

I made a simple implementation of the LDA model (topic modeling) for the “20 Newsgroup” dataset. It involves tokenization, stemming and lemmatizing, N-gram models, and gensim’s LDA model. The project and dataset can be found at:

https://github.com/ada5tra/jupyter

References

Sentiment Analysis Explained. Retrieved from https://www.lexalytics.com/technology/sentiment-analysis

Alice Zhao. (2018, July 28). Natural Language Processing in Python. Retrieved from https://youtu.be/xvqsFTUsOmc

Diego L. Y (2019, January 15). Your Guide to Natural Language Processing (NLP). Retrieved from https://towardsdatascience.com/your-guide-to-natural-language-processing-nlp-48ea2511f6e1

Joyce Xu. (2018, May 25) Topic Modeling with LSA, PLSA, LDA & lda2Vec. Retrieved from https://medium.com/nanonets/topic-modeling-with-lsa-psla-lda-and-lda2vec-555ff65b0b05

Hobson Lane, Cole Howard, Hannes Max Hapke. (2009) Natural Language Processing in Action: Understanding, analyzing, and generating text with Python. ©2019 by Manning Publications Co.

Aditya Jain, Gandhar Kulkarni, Vraj Shah.(2018). Natural Language Processing. Retrieved from https://www.researchgate.net/publication/325772303_Natural_Language_Processing

Selva Prabhakaran. Topic Modeling with Gensim (Python). Retrieved from: https://www.machinelearningplus.com/nlp/topic-modeling-gensim-python/

Victor Zhou. (2019, December 11). A Simple Explanation of the Bag-of-Words Model. Retrieved from https://towardsdatascience.com/a-simple-explanation-of-the-bag-of-words-model-b88fc4f4971

The Future of Natural Language Processing. https://www.youtube.com/watch?v=G5lmya6eKtc