Natural Language Understanding Labs

GitHub Repository

01_corpus_and_lexicon

Corpus is a collection of written or spoken texts that is used for language research. Units of Text Corpus
Depending on a goal, corpus can be seen as a sequence of:

- characters
- words (tokens)
- sentences
- paragraphs
- document
  Each level, in turn, can be seen as a sequence of elements of the previous level.

Lexicon is the *vocabulary* of a language. In linguistics, a lexicon is a language's inventory of lexemes.

Token vs Words: *tokens* are the elements in a sentences and they are used to compute the occurrences of a word. Instead, *words* are the unique elements that compose the Lexicon or Vocabulary of a corpus. We can think of words as classes and tokens as instances of those classes.

Frequency list: sorted list of words and their frequency.

- **Frequency Cut-Off** remove words that appear less/more that *N* times.
- **StopWord Removal:** stop words refers to the most common words in a language (they can vary dependening on the language).

Tokenization and Sentence Segmentation: given a "clean" text, in order to perform any analysis, we need to identify its units.
In other words, we need to *segment* the text into sentences and words.

02_experimental_methodology

Evaluation Metrics: *contingency table* is a type of table in a matrix format that displays the (multivariate) frequency distribution of the variables. For the binary classification into positive (*POS*)

and negative (*NEG*) classes, the predictions of a model (*HYP*, for hypotheses) with respect to the true labels (*REF*, for references) can be represented as the matrix.

- $\text{Accuracy} = \frac{\text{Num. of Correct Decisions}}{\text{Total Num. of Instances}}$
- Precision: how many selected items are relevant?
- Recall: how many relevant items are selected?
- F-Measure: harmonic mean of precision and recall.

Vectorizing Text: text classification requires vectorization, that converts text into a vector of numerical values. `scikit-learn` provides several vectorization methods, most commonly used are Count Vectorization and TF-IDF.

Bag-of-Words Representation: *count vectoriazation* implements the following vectorization procedure.

- *tokenizing* strings, i.e. splitting a string into tokens using for instance white-spaces and punctuation as token separators. Then, for each token it gives integer id.
- *counting* the occurrences of tokens in each document.
- *normalizing* and *weighting* with diminishing importance tokens that occur in the majority of samples / documents.
  Each token is considered to be a **feature** and the vector of all the token frequencies for a given document is considered a multivariate **sample**. Consequently, a corpus of documents is represented by a matrix with one row per document and one column per token (e.g. word) occurring in the corpus.

03_ngram_modeling

Frequency list of a corpus is a unigram count. **N-gram count** is a generalization of frequency list in which the length of the sequences to count is given by *N* instead of only 1 as default.

Tokens with frequency counts less than the `cutoff` value will be considered not part of the vocabulary, even though their entries in the count dictionary are preserved. This is useful for changing cut-off without recomputing counts.

In MLE LM we did not take care of OOV. Consequently, those have `0` counts and probabilities.

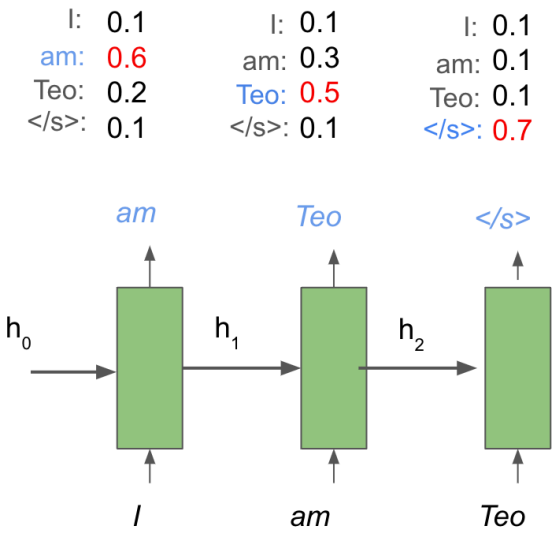
04_neural_LM

Word embedding is the representation of words in the form of a real-valued vector that encodes the meaning of the word such that the words that are closer in the vector space are expected to be similar in meaning. Word embeddings can be obtained using a set of language modeling and feature learning techniques where words or phrases from the vocabulary are mapped to vectors of real numbers. Conceptually it involves the mathematical embedding from space with many dimensions per word to a continuous vector space with a much lower dimension.

Pointwise Mutual Information is a measure of how often two events x and y occur, compared with what we would expect if they were independent.

Vector Similarity: Two words are similar in meaning if their context **vectors** are similar. **Cosine similarity** measures the similarity between two vectors of an **inner product space**. It is measured by the cosine of the angle between two vectors and determines whether two vectors are pointing in roughly the same direction.

Recurrent Neural Networks In the image below you can see a working example of a language model with RNN.



05_intent_and_slot_filling

Sequence labelling is to assign a label for each token. The task is formally defined as:

- Given a sequence of tokens $w = w_1, w_2, ..., w_n$,
- defining a sequence of labels as $l = l_1, l_2, ..., l_n$
- compute the sequence $\hat{l}$ such as $\hat{l} = \underset{l}{\operatorname{argmax}} P(l|w)$

A particular case of sequence labelling is **Shallow Parsing**. The main difference from Sequence Labeling task is that Shallow Parsing performs **chunking** -- segmentation of input sequence into constituents. Chunking is required to identify categories (or types) of *multi-word expressions*.
The **segmentation** part is represented with IOB tags and the **labeling** part are the concepts defined in the annotation schema of a corpus.

06_POS_and_NER

Sequence Labeling is a type of pattern recognition task that involves the algorithmic assignment of a categorical label to each member of a sequence of observed values. It is a sub-class of structured (output) learning, since we are predicting a *sequence* object rather than a discrete or real value predicted in classification problems.

The General Setting for Sequence Labeling

- Create **training** and **testing** sets by tagging a certain amount of text by hand
  - i.e. map each word in corpus to a tag
- Train tagging model to extract generalizations from the annotated **training** set
- Evaluate the trained tagging model on the annotated **testing** set
- Use the trained tagging model too annotate new texts

07_grammars_constituency

Parsing, syntax analysis, or syntactic analysis is the process of analyzing a string of symbols, either in natural language, computer languages or data structures, conforming to the rules of a formal grammar.
Within computational linguistics the term is used to refer to the formal analysis by a computer of a sentence or other string of words into its **constituents**, resulting in a **parse tree** showing their syntactic relation to each other.

A **Parse Tree** is an ordered, rooted tree that represents the syntactic structure of a string according to some *context-free grammar*. The term parse tree itself is used primarily in computational linguistics; in theoretical syntax, the term syntax tree is more common.

Context free Grammars CFG are defined by a *start symbol* and a set of *production rules*. The *start symbol* defines the root node of parse trees (usually **S**).
Production rules specify allowed parent-child relations in a parse tree. Each production specifies what node can be the parent of a particular set of children nodes.

For example, the production `S -> NP VP` specifies that an `S` node can be the parent of an `NP` node and a `VP` node.
The left-hand side of a production rules specifies potential *non-terminal* parent nodes; while right-hand side specifies list of allowed *non-terminal* and *terminal* (text) children.
A production like `VP -> V NP | VP PP` has a disjunction on the right-hand side, shown by the `|` and is an abbreviation for the two productions `VP -> V NP` and `VP -> VP PP`.

Probabilistic CFG are very similar to CFGs - they just have an additional probability for each production.

08_grammars_dependency

Unlike Constituency (Phrase Structure) Grammar that addresses how words and sequences of words combine to form constituents, Dependency Grammar addresses on how words relate to each other.

Evaluation of Dependency Parsing labeled and unlabeled attachment scores which are calculated as

$$UAS/LAS = \frac{\#\text{ of correct dependency relations}}{\#\text{ of dependency relations}}$$

the difference between the two is whether the relation labels are considered or not.

09_sentiment_analysis

Polarity Classification: classify an opinion exressed by a given text as positive, negative, or neutral.

Subjectivity/Objectivity Identification: classifying a given text (usually a sentence) into subjective (*opinion*) and objective (*factual*) classes.

Feature/Aspect-based Sentiment Analysis: sentiments expressed on different features or aspects of entities.