Introduction to Python for Social Science

Lecture 8 - Intro to Natural Language Processing

Musashi Harukawa, DPIR

8th Week Hilary 2020

This Week

Roadmap

This week we dive into my favourite field, *Natural Language Processing* (NLP), and look at the opportunities and difficulties that come with trying to harness text-as-data.

Points covered:

- What is NLP?
- Considerations about language as a data source
- Representations of Language and Relevant Metrics
- Application: POS-taggers, NER, and noun extraction.

Intro to Natural Language Processing

What is NLP?

Natural Language Processing (NLP) is an cross-disciplinary field drawing on linguistics, computer science, information retrieval, machine learning and artificial intelligence (among other fields) focused on computational methods for language. Applications include *speech recognition*, *natural language generation*, and *natural language understanding*.

- *Natural Language* is most easily defined in contrast to artificial or constructed languages. It includes all world languages, but excludes languages such as programming languages or Esperanto.

Social Science Applications

Applications in social science are usually focused on the *information retrieval/understanding* aspect of NLP. In other words, our focus is on using language as a data source, rather than building a working model for languages. As such, some major applications include:

- *Topic segmentation*
- *Summarisation*
- *Scaling*
- *Sentiment analysis*

Terminology

NLP has its own terminology, related to but separate from machine learning. Key terms include:

- *Text-as-data*: Applications of NLP focused on the extraction of information from textual data.
- *String*: A sequence of characters.
- *Token*: A string having a semantic function, often delineated by spaces in English. Otherwise a word, or a word fragment.
- *Document*: Sentences aggregated to a unit of analysis; can be a paragraph, a speech, a Tweet, etc.
- *Corpus*: A set of documents.
- *Vocabulary*: The set of all tokens. Usually the unique set of tokens contained within a given corpus.

Further Terminology

- *Lemmatization*: The reduction of a word to its grammatical root.
- *Stemming*: A set of rules for removing suffixes (and prefixes) from a word to reduce it to a word "stem".
- *Part-of-Speech Tagging*: The process of identifying the syntactic function of each token in a sentence, e.g. noun, verb, quantifier, etc.
- *Named Entity Recognition*: Automatic tagging of "named entities" within a text, usually proper nouns such as companies, people or places.

Language as a Data Source

Text-as-Data?

In general, as social scientists, we are not intrinsically interested in the process by which utterances and texts are generated (natural language generation) but rather the conditions which influence the generation of one text over another.

Put differently, we use language as a proxy to measure the states and processes that both produce and are produced by the data.

Example Question

- Imagine we are interested in the following question: *What is the effect of electoral system on legislators' Twitter usage?*
- How do we go about answering this question?

Formalizing Language

A tweet by legislator i of party j at time t can be described as an ordered set of tokens, drawn from the vocabulary V:

$$d_{ijt} := \langle w_1, w_2, ..., w_n, w_\omega \rangle \; \forall w_i \in V$$

Language Automata

Given this chain of tokens, we can describe the tokens as being probabilisitcally generated by an *automaton*, process which places probabilities over selecting one token given that another one came prior.

Generative Process

The probabilities in these automata can be described as a function placing probabilities over all tokens (the vocabulary) given the current state of the automaton. Aggregating this process up to the level of *document*, we can think of there as being a *document-generating process*.

The Tweet-Generating Process

$$d_{ijt} \leftarrow M(t, s_i, u_i, \lambda(\mathbf{w}_j, e_i, [...]), \mathcal{L}, [...])$$

Where:

- t is the substantive *topic* of the Tweet.
- s_i is the authorial *style* of legislator i.
- u_i is the preferences, or *position* of legislator i.
  - $\lambda()$ is a constraint function on the preferences of legislator i.
  - $\mathbf{w}_j$ is the aggregated preferences, or *position* of party j's elite.
  - e_j is the electoral formula faced by legislator i.
- $\mathcal{L}$ is the linguistic constraints on d, i.e. the language rules.

Sources of Variance

Each of these inputs and constraints contribute *variance* to the DGP. Lauderdale and Herzog (*PA*, 2016) provide a rough hierarchy of these sources of variance:

1. Language
2. Style
3. Topic
4. Position

Our Goal

As we are social scientists, and not computational linguists, we are usually not interested in *recreating* the document-generating process.

- Rather, we are usually interested in *quantifying* the effect of exogenous constraints on political processes.
- Thinking of text as a proxy for the states that produced those texts, we are interested in quantifying the effect of particular aspects of those states.
- Ultimately, our use case involves using statistical and machine learning methods to disentangle and quantify the heterogeneous sources of variance in our data.

Social Science Examples

- “A Scaling Model for Estimating Time-Series Party Positions from Texts”, Slapin and Proksch, AJPS 2008: Looks to infer party positions along a single ideological dimension with a model that assumes word counts are generated by a Poisson distribution.
- “How Words and Money Cultivate a Personal Vote: The Effect of Legislator Credit Claiming on Constituent Credit Allocation” (Grimmer, Messing and Westwood 2012)

Languages as Numbers

Representing Language

- We are already familiar with one computational representation of language: *strings*.
- All of the models we have encountered thus far require data that is both *numeric* and *tabular*.
- String data is *non-numeric* and *non-tabular*.
- Therefore we either need new kinds of models that can handle this structure of data, or a representation of language that can work with the kinds of models we have seen thus far.

Frequency-Based Approaches

The simplest numeric representation of language data is to simply count occurrences of words.

The document “The fox jumped over the fence.” can be represented:

the	fox	jumped	over	fence
2	1	1	1	1

Bag-of-Words

When we apply word counts to a *corpus*, we get a representation of language known as *bag-of-words*. Given two documents d_1 and d_2, "the fox jumped over the fence" and "the buffalo kicked the fence", we can write:

Document	buffalo	fence	fox	jumped	kicked	over	the
d_1	0	1	1	1	0	1	2
d_2	1	1	0	0	1	0	2

Variations

Some common variations of bag-of-words include:

- *Bag-of-ngrams*: An *ngram* is an ordered set of *n* words. Therefore a bag of bi-grams is the frequency of word pairs.
- *tf-idf*: Words are weighted by the product of two statistics:
  - *Term Frequency*: The frequency of terms in the document, same as bag-of-words.
  - *Inverse Document Frequency*: $log(\frac{n\ documents}{n\ documents\ containing\ w})$
  - This measure penalises words that occur across documents (e.g. "the", "and"), therefore favouring "unique" high-frequency words.

Limitations

- *Syntactic Information Loss*: Frequency-based approaches are purely *semantic* representations of language, and lose all *syntactic* information. i.e. they measure *what words people choose*, and not *what these words mean in conjunction* or *how these words are used*.
- *False equivalence*: Sentences containing the same words can have radically different meanings due to singular differences, or word order.
    - e.g. "The panda eats shoots and leaves" and "The panda eats, shoots and leaves".

Keep all this in mind when using models based on word frequencies— they often do not measure what you might think!

Vector Representation

Word and *document embeddings* refer to the vector representation of words and documents respectively.

Bag-of-words provides vector representations of words and documents in a corpus:

Document	buffalo	fence	fox	jumped	kicked	over	the
d_1	0	1	1	1	0	1	2
d_2	1	1	0	0	1	0	2

- $\mathit{vector}(d_1) = [0, 1, 1, 1, 0, 1, 2]$
- $\mathit{vector}(\mathit{buffalo}) = [0, 1]$

word2vec

Mikolov et al (2013) provide a method for word embeddings that is many senses superior to the bag-of-words model.

`word2vec` works by training a neural network to predict the middle word of a moving window of words through the corpus. The resultant vector space has some very attractive properties:

GloVe

Given the social sciences' (arguably justfied) aversion to neural networks, you may also prefer Pennington et al (2014), who take a more principled approach to word embeddings.

In essense, `GloVe` trains a statistical model to predict the co-occurrence of proximate terms. For further detail, see this excellent blog post.

In terms of accuracy, `GloVe` performs the same, or often *better* than `word2vec`, although the difference is limited for downstream analysis.

Applications for Word Embeddings

- As seen from the `word2vec` example, *distances* in the high-dimensional vector space generated by a model can accord with intuitive conceptual distances.
- With this space, we want to be able to *scale* or *cluster* words and documents, with the same intuitions that apply to other *scaling* and *clustering* tasks:
  - The ordering of and distance between observations on *latent dimensions* should correspond with the concept we are aiming to measure, such as ideology.
  - Words and documents classified as similar should be similar in some meaningful way, by topic, position, etc.

Measuring Similarity

- The standard metric for distance in Euclidean spaces, *Euclidean distance*.
- This is unsuitable for linguistic data for a number of reasons, the foremost being that word frequencies follow a *power law*.
- The distribution of word frequencies are highly skewed, meaning most of the distance between documents by magnitude will be the result of relatively meaningless differences in the occurrence of common words.

Cosine Similarity

- One common alternative to Euclidean distances is *cosine similarity*.
- The cosine similarity between two vectors is a function of the angle created by the two vectors from the origin.

Word Mover's Distance

Kusner et al (2015) apply Earth Mover's Distance, a measure of the minimum number of transformations from one distribution to another to sentences, calling this Word Mover's Distance.

Applications

spaCy

SpaCy provides industry-grade natural language processing resources with powerful out-of-the-box functionality. In particular, we can make quick use of the:

- Part-of-Speech Tagger
- Lemmatizer
- Named Entity Recognition

To a lesser extent, we may also be interested in:

- Document and Word Similarity
- Token Matching

Part-of-Speech Tagging

- `spaCy` provides a POS tagger that automatically labels tokens by their function within the sentence.
- We can use this to extract all of the nouns, adjectives, verbs, etc. from a text, a straightforward method for comparing intuitively a large collection of documents.

Lemmatization

- Often we do not care about the exact form of a word:
  - *investigate(s/d)*, *investigator(s)*, *investigation(s)*
- Lemmatization is the reduction of a word to its morphological stem.
  - *investigat*
- Stemming is similar, but uses a series of rules to simply cut off the ends of words.

Named Entity Recognition

NER is the automatic tagging of *named entities*. These include people, places, companies, events, and so on.

- This functionality can also be useful when combing through large quantities of data, but be aware that NER is less accurate than POS, and only recognises named entities that were in, or are similar to, the training data.

Things Not Covered Because They're Implemented in `R`

Structural Topic Model

Topic models are a class of dimensionality reduction model that discovers latent substantive “topics” in a corpus. They have powerful applications in classifying documents or measuring attention.

A dominant variant of the topic model is the *structural topic model*, which unfortunately is only implemented in `R`.

Wordfish and Wordshoals

Wordfish (Slapin and Proksch 2008) and its more recent context-aware variant Wordshoals (Lauderdale and Herzog 2016) provide *scaling* estimates of texts. Though originally applied to manifestos, it has a multitude of applications for quantifying one-dimensional variation between political actors.

References

Word Embedding Approaches

- "Concept Mover's Distance: measuring concept engagement via word embeddings in texts", Stoltz & Taylor (2019)

Topic Models

- "Structural Topic Model", Roberts et al (2015)
- "How Words and Money Cultivate a Personal Vote: The Effect of Legislator Credit Claiming on Constituent Credit Allocation", Grimmer, Messing and Westwood (2012)