Topic Models Part 1, Intro

Daniel Lawson — University of Bristol

Lecture 07.1.1 (v1.0.1)

Signposting

- This block is about Topic Modelling,
    - Bag of Words
    - Aside on Bayes
    - Latent Dirichlet Allocation

ILOs

Primarily:

- ILO2 Be able to use and apply basic machine learning tools

Bag-of-words model

- The bag-of-words model is the simplest tool for Natural Language Processing. It takes a trivial form:
  - A **vocabulary** is created, consisting of the set of all words in all considered documents.
  - Each **document** is represented as a **feature vector** by counting the number of occurrences of each word.
  - Typically, documents are **sparse** as most words do not appear in most documents.

Python Bag-of-words

```
import numpy as np
from sklearn.feature_extraction.text import CountVectorizer
count = CountVectorizer()
docs = np.array([
'The sun is shining',
'The weather is sweet',
'The sun is shining and the weather is sweet'
])
bag = count.fit_transform(docs)
```

- See Python Machine Learning[1].

[1] p259 Python Machine Learning (Raschka & Mirjalili, 2nd ed 2017).

Python Bag-of-words

```
>>> print(count.vocabulary_)
{'sweet': 4, 'shining': 2, 'weather': 6,
'and': 0, 'the': 5, 'is': 1, 'sun': 3}
>>> print(bag.toarray())
[[0 1 1 1 0 1 0]
 [0 1 0 0 1 1 1]
 [1 2 1 1 1 2 1]]
```

Word importance

- A popular measure of word relevancy is **term frequency-inverse document frequency** (**tf-idf**).
- If we call a document d and a word a "term" t, tf-idf takes a very simple form:

$$\mathrm{tf} - \mathrm{idf}(t, d) = \mathrm{tf}(t, d) \times \mathrm{idf}(t, d)$$

  - Where the term frequency $\mathrm{tf}(t, d)$ is simply the count of times term t is in document d.

- The inverse document frequency is:

$$\mathrm{idf} = \log\left(\frac{n_d}{1 + \mathrm{df}(d, t)}\right)$$

xw * Where n_d is the total number of documents,

  - $\mathrm{df}(d, t)$ is the number of documents d that contain the term t.

- Clearly this is arbitrary, though based on a reasonable principle.

Interpreting tf-idf

- idf can be interpreted as a (log) probability:

$$\mathrm{idf}(t, d) = -\log(\Pr(t|d))$$

- So the tf-idf is a bit like an information measure:

$$\mathrm{tf} - \mathrm{idf}(t, d) = -Pr(t|d)\log(\Pr(t|d))$$

Python tf-idf

```
from sklearn.feature_extraction.text import TfidfTransformer
tfidf = TfidfTransformer(use_idf=True,
                         norm='l2',
                         smooth_idf=True)
np.set_printoptions(precision=2)
print(tfidf.fit_transform(count.fit_transform(docs)).toarray())
[[ 0.    0.43  0.56  0.56  0.    0.43  0.  ]
 [ 0.    0.43  0.    0.    0.56  0.43  0.56]
 [ 0.4   0.48  0.31  0.31  0.31  0.48  0.31]]
```

Alternative transforms

- tf-idf is arbitrary. It induces a useful feature space for comparisons. It ignores word **usefulness**.
- Alternatives include:
  - Cosine Similarity
  - Any other transformation, especially those with information-theory interpretations
  - feature extraction methods to understand classification importance
  - **Word2Vec**: Implemented in the package `gensim`.
  - **Doc2Vec**: Another option.
  - Modelling, e.g. **Latent Dirichlet Allocation**.

N-grams

- The previous analysis treats words as a “unit of inference”.
- It is instead possible to consider **N-grams**, that is, all **occurrences of (up-to) N characters**.
- Given enough data, it is possible to learn the words.
- This is valuable for modelling, e.g.:
  - Foreign languages: all unicode characters can be handled,
  - Non-languages such as computer code or byte strings, such as seen in binary executables,
  - Arbitrary factor sequences.
- They are typically stored efficiently (see **hashing** later in the course).
- The penalty is that:
  - larger corpora are required to obtain the same classification performance,
  - the feature space is dramatically larger.

Reflection

Specifically:

- Know what a topic model is;
- Understand features in topic modelling;
- Understand the Bag-of-words model and how it can be used to compare documents;
- Know enough Bayesian Statistics to be able to understand what it does and does not do;
- Understand Latent Dirichlet Allocation at a high level.

Next Time

In the workshop we'll cover LDA in anger, with a focussed workshop session.

Bag of Words and Topic Models

Some references:

- Bag-of-words: p259 Python Machine Learning (Raschka & Mirjalili, 2nd ed 2017)
- Topic Modeling and Latent Dirichlet Allocation: An Overview (Weifeng Li, Sagar Samtani and Hsinchun Chen)