

Topic Models

Daniel Lawson — University of Bristol

Lecture 07.1.1 (v1.0.1)

Signposting

- ▶ This block is about modelling Languages, containing:
 - ▶ Part 1: The 'Bag of Words' model,
 - ▶ Part 2: An Aside on Bayes,
 - ▶ Part 3: 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 term (word).
 - ▶ Typically, documents are **sparse** as most words do not appear in most documents.

Notation

- ▶ **Terms** are indexed $t = 1 \dots T$
- ▶ **Documents** are indexed $d = 1 \dots D$
- ▶ **A document** X_d is a vector of term counts (sparsely stored)
- ▶ The **Corpus** $C = \{X_d\}_{d=1}^D$ is the set of all considered documents, and therefore contains all T terms

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 Learning¹.

¹p259 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)**.
- ▶ tf-idf takes a very simple form:

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

- ▶ Where the term frequency $\text{tf}(t, d) = X_d(t) / \sum_{t=1}^T X_d(t)$ is the frequency of term t in document d .
- ▶ The (log) inverse document frequency is:

$$\text{idf} = \log \left(\frac{D}{1 + n_d(t)} \right) = -\log \left(\frac{1 + n_d(t)}{D} \right)$$

- ▶ Where n is the total number of documents,
- ▶ $n_d(t) = \sum_{d=1}^n \mathcal{I}(X_d(t) > 0)$ is the number of documents d that contain the term t .
- ▶ The 1 is a smoothing term... (see Bayes in 7.1.2)

Interpreting tf-idf

- ▶ Clearly this is arbitrary, though based on a reasonable principle. . .
- ▶ TF accounts for the frequency within the document
- ▶ IDF assumes terms are *independent*, and ignores frequency:
 - ▶ The co-occurrence of two terms is the product of their probabilities, or the sum of their log probabilities
 - ▶ This ignores term frequency within each document
- ▶ This is therefore approximating $\Pr((t|d) \wedge (t \in d)) \log(\Pr(t \in d))$
- ▶ This can be rearranged into $\Pr(d|t) \propto \Pr(d, t)$,
- ▶ And resembles the elements of a **Mutual Information** measure:

$$(T, D) = \sum_t \sum_d p(t, d) \log \left(\frac{p(t, d)}{p(t)p(d)} \right).$$

Interpreting tf-idf

- ▶ The resemblance is meaningful, but not rigorous²
- ▶ Some hand-waving is required to get there:
 - ▶ $\text{tf} = \text{Pr}(t|d) \approx \frac{1+n_d(t)}{D}$ i.e. knowing the term tells you it is from one of the documents containing that term,
 - ▶ $\text{idf} = -\log(\text{Pr}(d|t))$
 - ▶ $\text{Pr}(d) = 1/D$
- ▶ The mutual information form can be reached by rearranging these sorts of statements
- ▶ It is not precise because different approximations are used in different elements
- ▶ And Mutual Information is a property of distributions, not of elements of that distribution.
- ▶ Very many other interpretations exist!
- ▶ These *hacks* can justified on robustness grounds.

²Stephen Robinson, Microsoft Research Understanding Inverse Document Frequency: On theoretical arguments for IDF

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,
 - ▶ word standardization cannot be used (see 7.2)

Reflection

- ▶ In tf-idf, how different is $\Pr(t|d)$ when using presence/absence, to using term frequency?
- ▶ What is a topic model mathematically? Can you distinguish between *instances* of a topic model, and what the general set of topic models looks like?
- ▶ What is a feature in topic modelling?
- ▶ What is good and/or bad about the Bag-of-words model?
- ▶ How would you quantify the loss of performance in an N-gram vs a language-aware model?
- ▶ How could you empirically compare topic models?

Signposting

- ▶ Bayes and LDA still to come in 7.1
- ▶ Practical considerations to come in 7.2
- ▶ In the workshop we'll cover LDA in anger, with a focussed workshop session.
- ▶ 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)
 - ▶ Stephen Robinson, Microsoft Research Understanding Inverse Document Frequency: On theoretical arguments for IDF