Chapter 1

Introduction

1.1 Numerical Representation of Language

Many modern machine learning techniques require a numerical representation of data and therefore cannot work with natural language directly. There are several different representations which have advantages and drawbacks. We will specifically look at vector space representations which are useful for classifying documents. This chapter will cover the simple bag-of-words model of documents and the more advanced word2vec model.

1.1.1 Bag-of-words

A bag-of-words model is a very simple numerical representation of a text document. The model assumes that a given document has already been tokenized. In bag-of-words, a single document is represented as the multiset of all tokens in the document with no regard toward ordering. This means that the conversion is, in general, not invertible and so the original document cannot be retrieved from this representation.

Consider the document

```
"This movie is not terrible, this movie is amazing!"
```

One possible bag-of-words representation for this document is

$$\{\texttt{this, movie, is, not, terrible, this, movie, is, amazing}\}$$

Notice that the above object is not a set, but a multiset since it contains one or more elements more than once. Also not that the following example is equivalent, since it contains the same words with the same frequency.

$$\{\texttt{this, movie, is, not, amazing, this, movie, is, terrible}\}$$

This object is conveniently represented as a hash table mapping tokens directly to their frequency in the document like the example in table 1.1

this	3
movie	3
is	3
not	1
terrible	1
amazing	1

Table 1.1: Hash table representation of a multiset

In the context of comparing two documents, it is sometimes more useful to represent a document as a vector. If we want to represent every possible document as a vector of the same length, then the number of elements must be equal to the total number of possible tokens, or the size of the vocabulary. For example, table 1.1 might be represented as

$$x = \begin{bmatrix} 0 & \cdots & 3 & \cdots & 3 & \cdots & 3 & \cdots & 1 & \cdots & 1 & \cdots & 1 & \cdots & 0 \end{bmatrix}^\mathsf{T} \tag{1.1}$$

with dots representing some amount of zeros. With this representation, one simple measure of document similarity is given by the *cosine similarity*, $s_\theta(d_1, d_2)$, of two document vectors

$$s_\theta(x_1, x_2) = \frac{x_1 \cdot x_2}{||x_1||_2||x_2||_2} \tag{1.2}$$

By the Cauchy–Schwarz inequality, this value has an upper bound of 1, which is achieved if and only if the two documents contain the same words with the same relative frequency.

The vector representation of documents also gives a hint as to how we might represent individual words numerically. If i is the index associated with the i^{th} word in a vocabulary, then we may associate that word with the standard basis vector e_i. Where every component of e_i is 0 except for the i^{th} which is 1. Let e_m be the standard basis vector associated with word m, then the vector represention for a document multiset, M, would be given by

$$x = \sum_{m \in M} v(m) e_m \tag{1.3}$$

where v is the multiplicity of element m in M.

In most cases, an actual language processing system will not use the raw frequency, $n_{t,d}$, of a term t and document d, but rather some term frequency function, $tf(t,d)$, of the raw frequency and given document which results in what is called a *term weighting*. Common choices are [1]:

1. $tf(t,d) = \{1 \text{ if } t \text{ appears in } d\text{; else } 0\}$
2. $tf(t,d) = n_{t,d}/N$, where N is the length of the document
3. $tf(t,d) = \{1 + \log(n_{t,d}) \text{ if } n_{t,d} > 0\text{; else } 0\}$
4. $tf(t,d) = \frac{1}{2} + \frac{1}{2}\frac{n_{t,d}}{N}$, where N is largest raw frequency in the document.

The inverse document frequency of a term is given by

$$idf(t,D) = -\log d_t/D \tag{1.4}$$

where d is the number of documents containing the term t, and D is the number of documents in total. One of the most common term weightings is then given by the term frequency-inverse document frequency (tf-idf):

$$tfidf(t,d,D) = tf(n,d) \times idf(t,D) \tag{1.5}$$

The bag-of-words model has the advantage of extreme simplicity, however it does not represent complex documents well since it has no regard for ordering. In addition, using this representation as a feature for machine learning is problematic since the number of features is equal to the vocabulary size which should be a very large number. We now look at latent semantic analysis, originally a method for indexing documents, which represents words and documents as vectors.

1.1.2 Latent Semantic Analysis

Latent semantic analysis (LSA) is essentially a dimensionality reduction technique similar to principle component analysis. At it's core is a truncated singular value decomposition (SVD) which is responsible for finding "key concepts" behind words. A set of word vectors for some vocabulary and set of documents can be found with the following steps:

1. Populate the term-document matrix, M: $M_{i,j} = tf(t_i, d_j)$
2. Using SVD, decompose M: $M = U\Sigma V^*$. U and V are unitary while Σ is diagonal.
3. Keep the largest k diagonal values in Σ, remove the other values' rows and columns and call the resulting matrix $\hat{\Sigma}$.
4. The vector corresponding to the i^{th} word is the i^{th} row of the matrix $U\hat{\Sigma}$

1.1.3 Word2vec

Word2vec is an extremely useful model which learns vector representations of individual words. Word vectors are learned in an unsupervised fashion, meaning that very large, unlabeled datasets can be leveraged during training.

Bibliography

[1] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval.* New York, NY, USA: Cambridge University Press, 2008.