Information retrieval

Francesco Tomaselli

March 10, 2021

Contents

1 Vector space model

Text transformation The first step to process queries on text is to transform it into a model that can be computed by a machine.

Such a model should be suitable for any kind of text, should be able to capture the semantic of words and then should support the notion of *text similarity.*

The main idea of a vector space model is to map documents in a vector, so they appear as points in a certain space.

For instance, we can collect some words and index them, then build a vector for each document

$$[x_0, \ldots, x_n]$$

Where the value x_i stores the occurrences of the word indexed with i in that document.

Similarity This model somehow implements the notion of text similarity, represented by some sort of distance function between two documents in space, perhaps the *Euclidean distance* or the difference between the angle of the two vectors. There are a lot of distance measures, and they vary in applications. Some of them assume normalization in the document vectors.

Document-Term matrix If we arrange document vectors in a vector we obtain the *document-term matrix*, or the *term-document matrix* if we take the opposite. The number of dimensions in the space is equal to the number of words in the vocabulary.
That can be a problem without some sort of lemmatisation and stemming, as sparsity and dimension of the matrix increases.

Vector length problem By considering the length of the vector in a similarity measure, we are introducing some problems, let's just consider a text and its abstract, the Euclidean distance would be huge, as most of the words do not appear that often.
To solve the problem we can consider the *Cosine similarity* or normalize vector lengths.

The main phases in the generation of a vector space models are: tokenization, normalization, weight, and indexing.

1.1 Tokenization

The goal is to split the text in words, for instance we could split the document at the spaces.

In practice this can be obtained with:

- *Regex*
- *Corpus* based
- *Machine* Learning based

1.2 Normalization

The idea is to normalize each token, that could mean writing verbs in the normal form, or plurals at the singular.

Stemming One form of normalization is stemming, that simply truncate words to obtain singular or infinite forms. The problems here could be creation of meaningless words and also close words in meaning could be completely different. An example is the *Porter* stemming algorithm.

Lemmatisation Another way to normalize is to lemmatise, It's similar to stemming, but the words are replaced with the dictionary root, this also has a problem, indeed sometime mapping of same words to different lemmas can happen. A way to lemmatise is to use a *Dictionary based* lemmatisation, or something more complicated such as *Wordnet*.

Wordnet Wordnet is a large lexical database, verbs, nouns, adjectives and adverbs are grouped into cognitive synonyms, called synsets, each expressing a different concept. Synsets are connected to each other by conceptual relations and they are also connected to a lemma.

1.3 Term Weighting

Weighting the words is a crucial point in text processing, an easy way could be counting the frequency of the terms in the document.

Term frequency An example of weighing is the term frequency, which counts the frequency of a term given a document:

- *Boolean*: so a 0 or 1 if the word is present or not
- *Natural*: $tf_{t,d}$ = count of a specific word
- *Log*: $1 + \log(tf_{t,d})$
- *Augmented*: $0.5 + \frac{0.5 tf_{t,d}}{\max_{t'} tf_{t',d}}$
- *Log average*: $0.5 + \frac{0.5 \log(tf_{t,d})}{1+\log(avg_{t'} tf_{t',d})}$
- *Max tf norm*: $k + (1-k)\frac{tf_{t,d}}{tf_{\max}(d)}$

Stop words problem A problem in this model is the excessive presence of stop words and punctuations. To compensate, we can remove them in the normalization phase.
But sometimes the stop words can be important, for instance with phrasal verbs. A solution can be to count the number of documents that contains that given word.

Inverse document frequency The idea is to count the number of documents that contains a given words, and to prefer less frequent words:

$$idf = \log \frac{N}{df_t}$$

There are many variants:

- *Smooth*: $\log(\frac{N}{1+df_t}) + 1$
- *Max*: $\log(\frac{\max_{t' \in d} df_t}{1+df_t}) + 1$
- *Probabilistic*: $\log \frac{N-df_t}{df_t}$

TfIdf By multyplying term frequency and inverse term frequency we obtain this metric.

$$TfIdf(t,d) = tf_{t,d} \cdot idf_t$$

Regarding the value of terms:

- the terms with a high *TfIdf* usually appears in a small number of documents
- the metric is low when a term appears a few times in a document or when it appears in many documents

2 Evaluation in information retrieval

The goal of evaluation is to assess the quality of the results obtained by an IR system. There's the need of knowing a *ground truth*, so an annotated corpus where for each task we know what documents are relevant. The annotations could be created manually or derived from the data, if it contains annotations.

2.1 The notion of quality

Given a corpus C and a query q, the task is to find a set of documents $A_{q,C}$ that match q, but after retrieving such documents, there's the need to estimate the quality of results.

Precision To formalise the quality of the retrieved documents $A_{q,C}$ we could count how many of these documents are relevant to q, this is the notion of precision:

$$Prec = \frac{relevant\ retrieved}{retrieved}$$

Note that in order to know if a document is relevant or not we need the ground truth or a user feedback.

This measure does not suffice, as for instance, if we retrieve only one correct document we would have maximum precision.

Recall The precision measure does not take in consideration how many relevant documents are there, that is crucial to assess quality of a query result. So we can consider another measure, called Recall:

$$Rec = \frac{relevan\ retrieved}{relevant}$$

Information need Given the two quality measures, should we aim at better precision or more recall? Trivially both, but it actually depends on the information need.
In some cases a really high recall is not necessary, while other times a lower precision could be accepted while not missing anything relevant.

F1 Score To take into consideration both precision and recall, we could take a weighted mean, so a tradeoff between the twos

$$F1 = \frac{2 \cdot Prec \cdot Rec}{Prec + Rec}$$

Baseline system To assess the quality of an IR system, we need a baseline system, something trivial such as tossing a coin to decide whereas a document is relevant or not. Given its quality measure, we can infer the quality of our, hopefully, more complex system, in fact, quality measures can't be seen as absolute values, there's always the need to compare.

2.2 Quality measures

To formally define the notions introduced in the previous section, we need to take into consideration a more detail measure for errors.

Confusion Matrix Given a query q, a ground truth E_q of relevant documents with respect to q, and a set of retrieved documents A_q, we define this matrix:

	$d \in E_q$	$d \notin E_q$
$d \in A_q$	True positive	False positive
$d \notin A_q$	False Negative	True Negative

So now we can redefine the *precision*, *recall* and *F1* measures as follows

$$\mathit{Prec} = \frac{TP}{TP+FP} \quad \mathit{Rec} = \frac{TP}{TP+FN} \quad \mathit{F1} = \frac{TP}{TP+\frac{1}{2}(FP+FN)}$$

The actual confusion matrix is obtained by counting the documents given the retrieved ones and the ground truth.

The matrix can be useful to estimate the system parameters, for instance, if the number of retrieved documents is set to a certain value k and the value of true positives is really high, probably a smaller k would do the job, while if a lot positives are left out, i.e, the matrix has a high number of false negatives, maybe an higher k would make the system better.

Other measures The are a lot of measures to estimate the quality of a system, such as *specificity*, *negative predictive value*, *miss-rate*, *fall-out* and *accuracy*. Finding the right measure is really important, for instance, using accuracy with an unbalanced dataset is not better than precision, recall and F1.

2.3 Evaluation of ranking systems

We talked about boolean retrieval systems, as we took into consideration if a document is retrieved or not.

In a raking scenario, we want to give importance not only to precision and recall, but also to the rank assigned by the system.

Setting a threshold One way to go is to set a specific threshold and compute the precision and recall of those top documents. This method does not take into consideration the ranking of the documents.

Precision at K If we take the first K retrieved documents, ordered by rank, we can estimate the quality of the rank by computing the precision for the top K documents. By iterating the threshold we can better estimate the performances, opposed to setting a unique threshold.
To decide what thresholds to use we could use the distribution of the system ranking.

Discounted cumulative gain This approach takes into consideration the ranking and discount the gain given by a document with respect to its position in it.

$$DCG = \sum_{i=1}^{n} \frac{R_j}{\log(i+1)}$$

where R_j is the rank assigned to a document.

Precision vs Recall curve Another approach is to compute precision and recall measures at each point in the ranking. If we take the measurements and plot them, we find a correlation between precision and recall and if we compute the integral of that function, we obtain the average precision.

To have a smoother curve, it's possible to interpolate the precision, so for each point in the recall axis, we take the maximum precision of consecutive points.