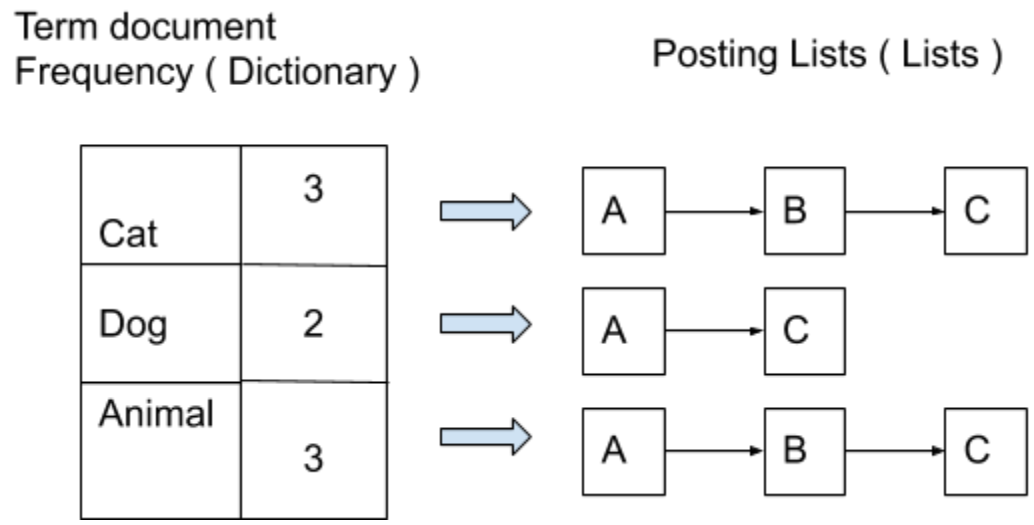
1)

Inverted Index : a given word, which document did it occur and how many times.

We need to create a data structure such that the terms point to the posting lists i.e. the list of all documents the term occurs in. This results in a dictionary data structure for the terms and list for the postings which is sorted by docID.

Sort alphabetically and sort by docIDs. The terms implemented using dictionaries include word count. Below is the Inverted index representation of the same.



2)

Text frequency :

	Cat	Dog	Animal
Doc A	0.571	0.285	0.142
Doc B	0.500	0	0.500
Doc C	0.142	0.428	0.428

Inverse Document frequency :

a) Cat : IDF =

$$= \log\left(\frac{total\ no.\ of\ \ documents}{total\ no.\ of\ \ documents\ with\ the\ word}\right)$$

$$= \log\frac{N}{n}$$

$$= \log\left(\frac{3}{3}\right)$$

= 0

b) Dog : IDF =

$$= \log\left(\frac{3}{2}\right)$$

= 0.176

c) Animal : IDF =

$$= \log\left(\frac{3}{3}\right)$$

= 0

TF-IDF Term frequency * Inverse Document frequency :

	Cat	Dog	Animal
Doc A	4*0	2*0.176	1*0
Doc B	3*0	0*0.176	3*0
Doc C	1*0	3*0.176	3*0

3) Documents A and C would be retrieved for the word dog.

4) Cosine similarity between query and documents.

$$Sim(x,y) = \frac{\sum_{i=1}^{n} x_i \cdot y_i}{\sum_{i=1}^{n} \left(\sqrt{(x_i)^2} \cdot \sqrt{(y_i)^2} \right)}$$

a) Sim (Cat , DocA) =

$$\frac{(4.4 + 2.3 + 1.1)}{\left(\sqrt{4^2 + 2^2 + 1^2}\right) . \left(\sqrt{4^2 + 3^2 + 1^2}\right)}$$

= 0.9843

b) Sim (Cat , DocB) =

$$\frac{(4.2 + 2.0 + 1.3)}{\left(\sqrt{4^2 + 2^2 + 1^2}\right) . \left(\sqrt{2^2 + 0^2 + 3^2}\right)}$$

= 0.7867

c) Sim (Cat , DocC) =

$$\frac{(4.1+2.3+1.3)}{\left(\sqrt{4^2+2^2+1^2}\right).\left(\sqrt{1^2+3^2+3^2}\right)}$$

= 0.6508

d) Sim (Dog , DocA) =

$$\frac{(3.4+0.3+3.1)}{\left(\sqrt{3^2+0^2+3^2}\right).\left(\sqrt{4^2+3^2+1^2}\right)}$$

= 0.6933

e) Sim (Dog , DocB) =

$$\frac{(3.2+0.0+3.3)}{\left(\sqrt{3^2+0^2+3^2}\right).\left(\sqrt{2^2+0^2+3^2}\right)}$$

= 0.9805

f) Sim (Dog , DocC) =

$$\frac{(3.1+0.3+3.3)}{\left(\sqrt{3^2+0^2+3^2}\right).\left(\sqrt{1^2+3^2+3^2}\right)}$$

= 0.6488

g) Sim (Animal , DocA) =

$$\frac{(1.4+3.3+3.1)}{\left(\sqrt{1^2+3^2+3^2}\right).\left(\sqrt{4^2+3^2+1^2}\right)}$$

= 0.7198

h) Sim (Animal , DocB) =

$$\frac{(1.2+3.0+3.3)}{\left(\sqrt{1^2+3^2+3^2}\right).\left(\sqrt{2^2+0^2+3^2}\right)}$$

= 0.6999

i) Sim (Animal , DocC) =

$$\frac{(1.1+3.3+3.3)}{\left(\sqrt{1^2+3^2+3^2}\right).\left(\sqrt{1^2+3^2+3^2}\right)}$$

= 1.0000

6) a) 0

b) To make the IDF finite, a word which doesn't occur in any document is discarded, i.e. we do not consider the terms whose term frequency is 0.

7) In contrast to the cosine, the dot product is proportional to the vector length. This is important because examples that appear very frequently in the training set (for example, popular YouTube videos) tend to have embedding vectors with large lengths. If you want to capture popularity, then choose dot product. However, the risk is that popular examples may skew the similarity metric. To balance this skew, you can raise the length to an exponent $\alpha < 1$ to calculate the dot product as $|a|^{\alpha}|b|^{\alpha}.\cos(\theta)$.

(Link : https://developers.google.com/machine-learning/clustering/similarity/measuring-similarity)

8)There is a good reason why accuracy is not an appropriate measure for information retrieval problems. In almost all circumstances, the data is extremely skewed: normally over 99.9% of the documents are in the non relevant category. A system tuned to maximize accuracy can appear to perform well by simply deeming all documents non relevant to all queries. Even if the system is quite good, trying to label some documents as relevant will almost always lead to a high rate of false positives. However, labeling all documents as non relevant is completely unsatisfying to an information retrieval system user. Users are always going to want to see some documents, and can be assumed to have a certain tolerance for seeing some false positives providing that they get some useful information. The measures of precision and recall concentrate the evaluation on the return of true positives, asking what percentage of the relevant documents have been found and how many false positives have also been returned.

(Link :
https://nlp.stanford.edu/IR-book/html/htmledition/evaluation-of-unranked-retrieval-sets-1.html)

9) For all $\alpha < 0.5$.

10) It quantifies the performance of the model as a whole, independent of any single query.

11) **Average precision at k** is the average of precisions across different values uptill k that correspond to the relevant documents.

$$MAP = \frac{\sum_{q=1}^{Q} AveP(q)}{Q}$$

Mean Average precision at k is the Mean of all average precision scores uptill k.

12) nDCG would be recommended due to the fact that Average precision is tailored for binary valued relevance judgements and cannot use a relevance scale, on the other hand since Cranfield dataset does contain, queries with relevance judgements, nDCG is preferred because it can handle relevance scales, and relevance judgements are not always binary but are sometimes on scale of one to X.

18) As the *n*-gram length increases, the amount of times you will see any given *n*-gram will decrease: In the most extreme example, if you have a corpus where the maximum document length is *n* tokens and you are looking for an *m*-gram where *m*=*n*+1, you will, of course, have no data points at all because it's simply not possible to have a sequence of that length in your data set.

The more sparse your data set, the worse you can model it. For this reason, despite that a higher-order *n*-gram model, in theory, contains more information about a word's context, it cannot easily generalize to other data sets (known as overfitting) because the number of events (i.e. *n*-grams) it has seen during training becomes progressively less as *n* increases.

On the other hand, a lower-order model (unigram) lacks contextual information and so may underfit your data.

Assuming that you have enough data to avoid overfitting, you then get better separability of your data with a higher-order model.

(Link:https://stackoverflow.com/questions/36542993/when-are-uni-grams-more-suitable-than-bi-grams-or-higher-n-grams)