# Vector space representation and similarity

Normalisation

Let’s have an n-dimensional space called Vector Space. This model is given from a collections of vectors, fundamental for operations in information retrieval, document classification, clustering and several others NLP applications. This vector space may be a representation of a document collection.

Premises: tf-idf is the weighting scheme of choice for this system but this space model may apply to any different scheme. Anyway here the assumption is the adoption of tf-idf.

This space has one axis for each term. The ordering of the terms is lost in a Bag of Words representation.

One way to measure the similarity between two vectors is to consider the difference in magnitude of two vectors. That can have a huge drawback in case two vectors, although extremely similar, have a considering difference in length. That may cause one to have a much greater absolute term frequency.

To deal with the effects caused by the above situation, one common method to calculate the similarity is to obtain the *Cosine Similarity* between the two vectors:

Description: Numerator being the *Dot Product* of the tow vectors, denominator being the product of *Euclidean Lengths* defined as , where being the vector representation of a document and *M* components.

The *Euclidean Lengths* has the effect to normalise the length of two vectors in relation to each other. It could be not essential for already normalised collections of vectors (like as through the use of normalising algorithm for weighting such as TFIDF).

Dot Product is the product of each term to the corresponding one (same position or same term) in the other vector:

This is done to assert the effective similarity of said documents, relating them on a imaginary space having a centre, which each vector being a line starting from that (0) forming angles. The geometric angle between two vector will be the Cosine (θ), which value can be the degree formed by this new “shape”. Analysing all degrees between all vectors, cosine values can be used in different fashion, depending on the scheme for the clustering (as a choice for this project, but can be for many different uses in information retrieval, summarisation, etc…). Example, they can be used in descending order for Single-Link scheme.

D2

D3

D1

 ϴ

Vectors themselves abstract well beyond 3-dimensions, up to the number of components forming a vector.

Vectors can be seen as special matrices with only one column.

Dot Product will result: a1b1+a2+b2,…an+bn giving out a True Scalar.

DOT PRODUCT is COMMUTATIVE: the order the vectors are used does not matter.

DOT PRODUCT is DISTRIBUTIVE: let’s have a vector from the addition of other 2 vectors [v1+w1,v2+w2,….vn + wn]. Dot product with another vector [x1,x2,….xn] should satisfy distribution by rendering as true (v1+w1)x1 = x1+v1 \* x1+w1.

Vector length: the dot product of a vector V with itself (V \* V) can be seen also as the sum of each component squared. In case of a document vector, each term’s weight squared.

Looking at Pythagora, the theorem recites that the distance from a centre point 0

For A = (a1, a2, ..., an), the dot product A**.**A is simply the sum of squares of each entry.

In the plane or 3-space, the Pythagorean theorem tells us that the distance from O to A, which we think of as the length of vector OA, (or just length of A), is the square root of this number.

**Definition**. For A in n-space, the length of A = square root of A**.**A. This length is written |A|. so |A|2 = A**.**A.

Likewise, the Pythagorean theorem also shows that the distance from A to B is the length of AB, which is the length of B-A.

**Definition**: For A and B in n-space, the distance from A to B is the length |B-A|.   
Note: This equals |A-B| also.

### Law of Cosines

**Cosine Theorem:**For A and B in the plane or in space, A**.**B = |A| |B| cos AOB

In the triangle AOB, if we let the length of side AB = c, then by definition of distance,

c2= |A - B|2

Proof: From the algebraic properties,

|A - B|2 = A**.**B - B**.**A - B**.**B = |A|2 + |B|2 - 2A**.**B.

But from geometry, since |A| = |OA| = side opposite B, etc., the Law of Cosines for triangle AOB says

c2= |A|2 + |B|2 - 2|A| |B| cos AOB.

Comparing these, all but one term is the same in each, so we see that A**.**B = |A| |B| cos AOB