MapGeist - The Basics

Sachin Joglekar

1 Introduction

As a Python library/Mind-mapping tool, MapGeist aims to exploit concepts from the domains of **Text Mining** and **Graph Theory** to provide a comprehensive means of understanding/exploring text graphically.

The following sections describe the basic algorithms behind MapGeist's working (as of now).

2 Methodology

2.1 Step 1: Computing the top N n-grams

The first step towards building the MapGeist Mind-Map is to figure out the top N n-grams from the text, where N is provided by the user. This is done using the Single-Document Keyword Extraction algorithm proposed in [1]. [1] allows us to quantify the *importance* of each n-gram in the document. Intuitively, if a k-gram is a *part* of an m-gram, where k < m, and $importance(k) < importance(m)$, then the k-gram is taken out of consideration and the next most important n-gram takes its place in the ranking.

2.2 Step 2: Building the distance matrix

Once the top N n-grams have been figured out, an ordering is defined for them, and a co-occurence matrix is constructed. Let this matrix be called C. Therefore, $C[i][j]$ denotes the number of times the ith and jth n-grams appear together in sentences throughout the text. Now, for the ith n-gram, its *scaled* co-occurence $C_s[i][j]$ with the jth n-gram can be defined as:

$$C_s[i][j] = \frac{C[i][j]}{\sum_k C[i][k]} \tag{1}$$

Using these scaled co-occurences, the normalized co-occurence matrix C_{norm} can be defined as:

$$C_{norm}[i][j] = C_{norm}[j][i] = C_s[i][j] * C_s[j][i] \tag{2}$$

By definition, C_{norm} is a symmetric matrix. The main advantage of the co-occurence quantification in C_{norm} (over that in the original matrix C) is that it's independent of *how many* times a word occurs in the document. As a result, words that occur extensively in the document do not have a high 'affinity' towards all the words in it. $C_{norm}[i][j]$ will be high iff n-gram i is one of the top n-grams j co-occurs with, and vice versa. Thus, C_{norm} is a much better estimation of the affinity between terms in the text, rather than C.

Using C_{norm}, a distance-matrix D can now be constructed with respect to the top N n-grams, using the following definition:

$$D[i][j] = \frac{1}{C_{norm}[i][j]} * \frac{\|V_{i,j}\|\|V_{j,i}\|}{(V_{i,j} \ \cdot \ V_{j,i})} \tag{3}$$

if $i \neq j$, where $V_{i,j}$ is a vector constructed by combining the normalized co-occurences of n-gram i with all n-grams except i and j. The above definition of the distance between two different n-grams takes into account not just their co-occurence, but *how similar* are their co-occurences with other n-grams in the text. Ofcourse, the distance of an n-gram from itself is zero:

$$D[i][i] = 0 \tag{4}$$

Once D is built, we can then move on to constructing the actual Mind-Map using Graph Theory.

2.3 Building the Mind-Map from the distance matrix

Once the distance matrix is built, we construct a complete graph consisting of the top N n-grams from Step 1. The distances between each set of n-grams is defined by the distance matrix that was derived in Step 2.

Every Mind-Map can be thought of as a Tree. Intuitively, if a Mind-Map were to be constructed from the complete graph we just talked about, there would be only ONE way to reach any node (n-gram) from another node. Moreover, the structure would be optimal in the sense that the on an average, the length of such a distance between any two n-grams would be minimal. Therefore, to generate the 'optimal' tree using the distance matrix we built in the previous step, we compute the **Minimum spanning tree** corresponding to the aforementioned complete graph.

3 References

[1] Keyword extraction from a Single Document using Word Co-occurrence Statistical Information - Y Matsuo, M Ishizuka (FLAIRS 2003)