APPM4720/5720 Project Proposal

Rishabh Raghavendran - William Farmer

March 23, 2016

1. Introduction:
   (a) Overview:
   Finding the meaning of documents and drawing relations between different documents is an extremely human activity. We as humans possess the innate ability to reason and make associations based on this. In today's age of automation and artificial intelligence, is it possible to replicate this behaviour using a computer to any extent? We believe it is and the aim of our project is to use some smart mathematics to do exactly that. But before we move forth we need to actually decide whether practicality to doing this. Why would we even want the computer to replicate this sort of human behavior? Aside from scholastic and academic curiosity about the challenge in solving this problem, there are bona fide reasons for doing this; the primary one being that computers are much better than humans at doing repetitive tasks many times. While it would take a human many days to go through a large set of documents, a computer would theoretically be able to do that in a fraction of the time.

   (b) Our proposed Solution:
   To solve the aforementioned problem, we want to perform Latent Semantic Analysis using the Singular Value Decomposition of the documents and the associated document-term matrices. Our preliminary research has shown that by using this algorithm, we can not only query for certain keywords in documents, but also find the documents that are in some way related to the query word but do not explicitly contain it.[1]
   The question is, can a Machine do this? The answer is Yes, LSA does exactly that.

[1] For example, suppose we have the following set of five documents and a search query: dies, dagger.

d1	Romeo and Juliet.
d2	Juliet: O happy dagger!
d3	Romeo died by dagger.
d4	“Live free or die”, that's the New-Hampshire's motto.
d5	Did you know, New-Hampshire is in New-England.

Clearly, d3 should be ranked top of the list since it contains both dies, dagger. Then, d2 and d4 should follow, each containing a word of the query. However, what about d1 and d5? Should they be returned as possibly interesting results to this query? As humans we know that d1 is quite related to the query. On the other hand, d5 is not so much related to the query. Thus, we would like d1 but not d5, or differently said, we want d1 to be ranked higher than d5.

   (c) Our Dataset and Metric for success
   So far we have come up with an interesting problem to solve and an ingenious technique to solve it, but are there any real world datasets for which this problem is a viable one? In fact, the internet is full of these kind of datasets and for this project we will explore a few of the ones we find most interesting. For example, we can obtain a dataset of all the recipes in a book and then query for specific ingredients or cuisines. For example, suppose we wish to make some thai food with chicken, rice and a few other ingredients, we can search the entire database of recipes and arrange them according to what matches perfectly with our query, and others which are not exactly that but fairly similar.
   Another interesting dataset for which this problem would yield some interesting results is the database of the inaugural addresses of each of the presidents of the USA to find what are the words and related topics most spoken about by them and draw comparisons between their speeches and the speeches of other famous people.

2. Latent Semantic Analysis using SVDs:
   (a) What is it?
   LSA (or sometimes referred to as Latent Semantic Indexing (LSI)), is a method to discover hidden concepts in document data. It does this by analyzing relationships between a set of documents and the terms they contain by producing a set of concepts related to the documents and terms.
   LSA presumes that words that are close in meaning will occur in similar pieces of text.
   (b) Implementation
   First, we will construct a document-term matrix from a large piece of text. We will then perform dimensional reduction using singular value decomposition so as to reduce the number of rows while preserving the structural similarity among columns. Words are represented as normalised vectors and are compared by taking the dot product between them. We wish to implement this in Python. The reasons for that are:
      i. Ease of implementation
      ii. A vast library of resources
      iii. Author Familiarity
3. Additional Techniques and Modifications:
   Besides the aforementioned techniques using LSA, we also will explore other techniques for large-scale document analysis and natural language processing. Note, some of these extensions and modifications are intertwined and rely on many of the same core concepts. In this section we will also discuss the qualities of the different methods versus our initial LSA implementation.
   (a) Cluster analysis and Non-negative matrix factorization
   Using matrix factorization of the document-term matrix we can factor and establish clusters. Analysis can then be performed on these clusters for similarity and term searching.
   (b) Probabilistic Latent Semantic Analysis

An extended version of LSA, PLSA uses multinomial distributions to model the behavior of our documents using a multinomial distribution.

(c) Other extensions
As we perform our research, we will undoubtedly come across several other well-studied methods to perform this type of natural language processing, and they may be added at a later time.