Catagory Clustering for Query Classification*

David Blackford
Victoria University of Wellington
Kelburn Parade
Wellington
New Zealand

Abstract

Short Query classification is an work area that relies on the enrichment of short queries in order to get enough information to successfully classify the query into one of many catagories. This task can be done in many ways, and this paper discusses my attempt at a clustering algorithm designed to catagorize a query.

Algorithm

The algorithm that I designed is based on the work (**?**) and attempts to find the most common distribution of catagories for words that are related to a given query. The query begins unenriched, and is enriched through use of the Google search engine(http://www.google.com). The resulting body of words is taken as related to the query, and these words are used to find a distibution of the algorithms relative belief that the query should belong to a given catagory/catagories. Each word for the enriched query measured against each of the given catagories using one of many possible measures. Once we have measured the rough probability distribution of the word belonging to the different catagories, we have to decide on which of these distributions comes closest to representing the query which the words are enriching.

In this algorithm we do so by, rather than selecting a distribution, by clustering similar distributions until one distribution becomes a fair representation of the majority of the distributions we began with. This is completed throught use of the Kullbeck-Leibler algorithm (**?**) which is an algorithm which calculates a measure, the dissimilarity, between two given distributions. If two distributions are similar, then the dissimilarity value returned by the Kullbeck-Leibler algorithm will be extremely low. If two distributions are completely unlike each other, then the dissimilarity values in both directions will be low. One issue with the Kullbeck-Leibler algorithm is that it is a bi-directional algorithm, that returns different values in each directions in most cases. It is described in the work by (**?**) as an algorithm which describes how well messages intended for one distribution fare when they are sent to the other distribution, which is a process which will be different depending on which distribution the messages were originally intended for. This leads to different values depending on the order of the distributions. In this paper they work around this issue by averaging the message based on how often a word appears in the category. In the algorithm that I am working on I have gone with a much simpler pure average, much like the one used in their earlier paper (**?**)

Implementation

Written in java etc The measure that was originally attempted for this algorithm was the relatedness measure found from Wikiminer (**?**) however the implementation of the Wikiminer program was poorly documented, and the relatedness measure relied on a webservice that seemed to be shut-down. Another measure that was attempted was use of Normalized Google Distance (**?**). This is a measure which makes use of the Google search engine's approximate pages in order to calculate how often two words appear together relative to how often they appear apart. This measure seems extremely promising, however Google places limits on the number of queries and frequency of queries. For initial

*Thanks to Xiaoxing Gao for your help