Category 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 categories. This task can be done in many ways, and this paper discusses my attempt at a clustering algorithm designed to categorize a Query based on the results of search engines.

Introduction

The amount of information that we have at our disposal when we connect to the Internet is growing, and has been since the Internet's inception. With the amount of information that is now available, Web search and the algorithms surrounding Web search is now more important than ever. These Web search engines allow users to submit short queries for information and be presented with information that is related and relevant to their needs. Due to the length of these short queries, the majority of which are less than 3 words (Hölscher and Strube 2000), this becomes a more complicated problem due the two core issues. Short queries only provide the search engine with a very small set of words with which to select relevant information, and these ambiguous words could possibly have a range of different meanings. Query Enrichment is a solution to the issue that there are only a few words, using the words to build a body of knowledge surrounding the words, rather than simply relying on the occurrence of the words in different result documents. Query Categorization aims to provide better search result pages for users with interest in different categories, however is not a trivial task due to the short, noisy and ambiguous nature of the queries. The KDDCUP 2005 (http://www.acm.org/sigkdd/kddcup) is a competition where programmers are tasked with categorizing 800,000 web queries into 67 target categories, each Query being able to belong to more than one category. It is this competition which has led to the creation of some of the algorithms discussed in this paper, and it is on this competition that this project is intends to emulate. The improvement of Search algorithms itself is a growing source of information, and for each Query it is possible to reach many possibly related results. For each result though we have the need to decide which categories these results belong to, and if we arrive at a common set of categories for the results, can we come to the conclusion that the Query too is related to these categories. For the last few weeks I have been working on creating an implementation of an algorithm that will categorize short queries using clustering of related words to make judgements on the category that each Query belongs to.

[*]Thanks to Xiaoxing Gao for your help, It was much appreciated

Related Work

Web Classification has a wide variety of different algorithms attempting to improve the way we classify queries. Some methods attempt to cluster user behaviour as opposed to the queries specifically, using user logs to enrich their knowledge of which queries are related to which results, and also which queries are related to each other, which in turn tells you which results are related to each other (Wen, Nie, and Zhang 2001). This particular clustering algorithm had the goal of finding Frequently Asked Queries using these user logs, and from these Frequently asked Queries collated the frequently accepted results. Like many algorithms it used a hybrid of many different types of similarity measures, focusing on Query contents and user feedback from the user logs. In the past there have been many clustering algorithms created in the area of Text Classification. Many like those discussed above, use information collated over time and past experiences, others cluster queries by their semantic relatedness or a distance calculated from Wikipedia or other encyclopaedias (Hu et al. 2009). The clustering itself ranges from Hierarchical to K-Means (Hartigan and Wong 1979; Hu et al. 2009) to DB-SCAN (Wen, Nie, and Zhang 2001). The general goal was to find allow unknown queries to become known based on previously categorized queries, or to cluster the queries together so that informed approximations could be made about the entire cluster. (Baker and McCallum 1998) discussed word clustering, stating that word clustering allowed (1) useful semantic word clusterings, (2) higher classification accuracy and (3) smaller classification models. Pereira et. al (Pereira, Tishby, and Lee 1993) created an algorithm based on the Kullback-Leibler(KL) Algorithm (Kullback 1987) that attempted to simply cluster words from the English language into different different senses, or underlying meanings using a combination of hierarchy and distributional clustering. This idea was later selected and improved by (Baker and McCallum 1998) in an attempt to enrich short queries into goal categories. This algorithm is reasonably simple, once the KL algorithm is understood.

$$\sum_{j=1}^{|C|} P(c_j \mid \omega_t) \log(\frac{P(c_j \mid \omega_t)}{P(c_j \mid \omega_s)}) \tag{1}$$

Equation 1 shows the KL algorithm calculating the KL Divergence between the category distributions induced by word ω_s and ω_t and is written $D(P(C \mid \omega_s) \parallel P(C \mid \omega_t))$.

KL divergence has some strange properties. It is not symmetric, and it is infinite when confronted with one zero and one non-zero probability. It is often used to find the distances between two probabilistic models, and the algorithm uses it in order to find the dissimilarities and similarities between different categories and the short queries.

Algorithm

The algorithm designed for this paper is based on the work (Baker and McCallum 1998) and attempts to find the most common distribution of categories for words that are related to a given Query. The Query begins un-enriched, and is enriched through use of the Google search engine (http://www.google.com). The resulting body of words is assumed to be related to the Query in some form, and so these words are used to find a distribution of the algorithms relative belief that the Query should belong to a given category/categories.

The intended benefits of this approach is an attempt to determine the category of the Query by the majority vote of the related words. This does not occur as is discussed in the Experimentation section, however this is mainly because many of the words returned are generic words used in all categories and should possibly be included in stopwords, with only a few specific words carrying a large probability for any given category.

Each word for the enriched Query measured against each of the given categories using one of many possible measures. Once we have measured the rough probability distribution of the word belonging to the different categories, 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, clustering similar distributions until one distribution becomes a fair representation of the majority of the distributions we began with. This is completed through use of the Kullback-Leibler (KL) algorithm (Kullback 1987) 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 Kullback-Leibler algorithm will be extremely low. If two distributions are completely unlike each other, then the dissimilarity values in both directions will be low. The KL algorithm has been used in may different research projects for comparison and clustering of words in language, (Pereira, Tishby, and Lee 1993)

Once each word has a probability distribution, the strongest distributions were selected to seed the clusters. The

1. Enrich Query Q to populate word corpus W
2. For each word w in W calculate distribution $P(C|w)$ for Categories C
3. Add M words as seed clusters where M is the number of goal clusters
4. Compare next word w to each cluster c with KL(algorithm 1)
5. merge w into closest cluster, averaging the distribution of w and c
6. Repeat from step 4 until there are no more new words...
7. Set the distribution of Q to the strongest distribution amongst the M clusters

measurement and effects of this are discussed in the experiment section, though the intention of this was to ensure that there were strong contenders for the final merged distribution already in the results, so that the result set did not immediately fill with noise words. After selecting M initial cluster seeds from the list of distributions each distribution representing a word's probability to belong to the categories is from the remaining list and compared with each of the cluster seeds using KL. The distribution is merged into the cluster that it was most similar to resulting in a cluster containing an averaged distribution. This continues until there are no more distributions left in the initial list, and the strongest resulting cluster is selected to represent the entire corpus and therefore, the Query. The distribution at the end will contain the merged distribution of all of the most promising words from the enrichment, and so if one takes the words as body to be related to the Query, then the merged cluster is representative of the Query.

Issues with the algorithm design

One issue with the Kullback-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 (Baker and McCallum 1998) 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 the earlier paper (Pereira, Tishby, and Lee 1993). I have avoided using the more variably weighted averaging due to the knowledge that at least for now, the measures used to measure the probability of a single word in this dataset aren't based over the whole dataset, focussing instead on the measurement within a cluster's word corpus. I feel that leaving the simple average in will cause less assumptions to be made on the importance of any given word, instead allowing only the word's importance in relation to the category to affect the result.

Despite this there are still issues with using the Kullback-Leiber algorithm. For example, when one distribution's category has a very low probability, and the other distribution has a very high probability, the dissimilarity reaches towards infinity, which is most prominent as an issue when one distribution has a probability of zero for a particular category and the other does not. This case, where one value is zero and one is not, is an occurrence which can't be over looked. In these cases the dissimilarity will be infinite, an issue which (Baker and McCallum 1998) didn't discuss their solution to. (Pereira, Tishby, and Lee 1993) did discuss this problem but came to the conclusion that because they were working with the comparison between a word that they know has non-zero probability and a centroid cluster they would never come across this issue, as the centroid would have a non-zero result as long as a word contained within did. In my case, I ensure that there is a moderate penalty value attributed to this which will be difficult but not impossible to overcome in order to become a distribution others agree with, but only by having less of these situations occurring than the other. This change may be one that cripples the algorithm, causing any cases where this occurs to lean too heavily towards not having any points where one is zero and the other is not. If the category for which this occurs is completely unrelated to the document as a whole, and possibly disagree with the rest of the distribution, it could cause that distribution to not pair with any of the other distribution, whether they too are unlikely to be placed in that category or not. This however was a situation where it is possible that the only realistic solution was ensuring that there was some, possibly minuscule probability of being any given category. Balancing this could become a major problem, in either solution, but it is something that would need to be considered in order to improve this algorithm further.

Implementation

The implementation of the Query clustering algorithm was written in Java and made use of Bing(www.bing.com) as a search engine service to compare Word Enrichments to Categories. The words were originally enriched using The measure that was originally attempted for this algorithm was the relatedness measure found from Wikiminer (Witten and Milne 2008) however the implementation of the Wikiminer program was poorly documented, and the relatedness measure relied on a web-service that seemed to be shut down. Another measure that was attempted was use of Normalized Google Distance(NGD) (Cilibrasi and Vitanyi 2007). 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. As a proof of concept though I feel that the NGD distance would be sufficient, and show the merits of the distribution clustering algorithm. Given the earlier discussed issues with the algorithm, NGD would only result in problems when one of the enrichment's returns 0 results when searched with one of the categories. This would result in the Kullback-Leibler algorithm returning an infinite dissimilarity between it and any other distribution. To avoid this situation, in the case where there is a 0 probability for one of the enrichments belonging to a category, rather than resulting in an infinite dissimilarity, I limit the dissimilarity to a much smaller (than infinity) penalty value. This still results in an extremely high dissimilarity, however it no longer causes issues in the comparison of distributions and clustering.

One of the important parts of this implementation is that all data is stored given the chance, so once it has been worked out once, it is simply a case of accessing the data again rather than recalculating. This improves the computation time of the algorithm immensely, but also ensures that it will scale acceptably when run on larger datasets.

Seeding the initial clusters soon became an issue, one that I found I would be unable to solve until I found a different way to organise the addition of new Words and the management of existing Clusters. The issue was that once a word joined a Cluster, even if it only joined it because it was being used as a seed, that word can no longer leave the cluster and the cluster is unable to merge with existing clusters. Solutions to this included allowing a grace period upon completion of the algorithm for the least cohesive distributions to disperse and rejoin the system, as well as allowing cross merging between existing clusters, however managing the number of clusters that would be in the result became an immediate issue. Allowing merger and dispersion of clusters could easily result in loops where alternative options are less desirable than continuously merging and dispersing. I decided that there were other solutions that were more pressing and would have a stronger effect, however this solution is an option for improving the results of the algorithm if it can be completed. Following in (Baker and McCallum 1998)'s footsteps by creating M clusters, merging two and filling the spot with a non-clustered word could also have more success and could be easily implemented as I suspect that having these seeds for clusters and merging new rather than existing clusters is causing too many issues.

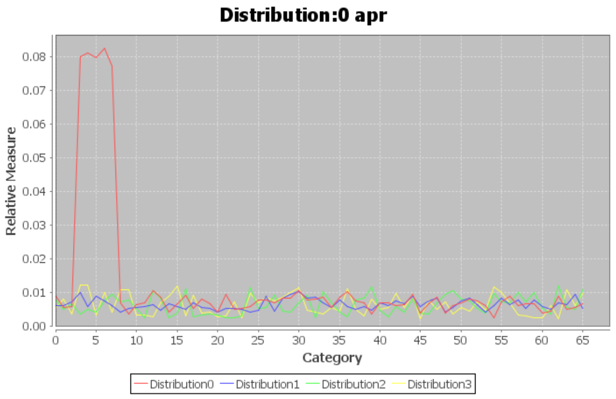
Experimentation

While I was able to get the algorithm working, I am as of yet unsure of it's success. Because of issues discussed earlier in finding a rich body of enrichment and measurement for each Query and related words, I feel that while my algorithm shows promising results on the individual runs I have completed for different categories, I do not have a convincing proof of the algorithm's strengths.

The individual runs I have completed on 2 categories run through the algorithm, despite a large amount of remaining noise in the dataset, were reasonably successful.

Issues that I found in implementation were that it became a problem with not knowing how to select the cluster that I was going to choose to represent the Query. as an example, my experimentation on the Query "0 apr" resulted in two clusters containing only single words. These words were "apr" and "credit" so are extremely related to the Query, however they were so distinct in their levels of relation that few other words or clusters deemed them close enough to cluster to. My initial reaction is to use the distribution attributed to the word with the highest average distribution probability,This simple algorithm for selecting the ”strongest” distribution could have many more issues when dealing with clusters with very broad but moderately thick distributions, which could have a much higher average despite the words contained within having only very broad and generic category preferences. This however may have been a side effect of these two words being selected initially as the seeds of the categories, so while they were related to each other they were never able to be merged into the same category. In later revisions a spread was selected from the distributions, rather than the strongest, as selecting the strongest resulted in singleton clusters. These singleton clusters with very few results or single words often occurred because the only other words or clusters that were similar enough to merge with were already forced into other clusters as the algorithm initiated. Having a spread of distribution strengths caused a reasonable spread of results, though they were still often heavily based on the initial starting distributions. Two clusters that have at least mildly different strengths are less likely to later on require being in the same cluster. ”Credit” and [”Apr”, ”Card”] for example could be found in the same results cluster when this change was in place, in comparison with choosing the strongest distributions as the seed for the results

When implemented with a cluster seed that spread the possibly related words out (by selecting differing strength distributions) the resulting clusters contained a much more informative and robust result. While there were two major clusters in previous runs, both for the same general set of categories (with some differences) there was now one cluster containing [”Credit, ”Apr”, ”Card”, ”Finance”], the most words with the strongest relation to a category (see Distribution0 in the graph).



One of the other clusters in particular held the majority of the remaining results, with most of the unrelated words grouping together due to common unrelated-ness to the categories. I would like to look into how it would be possible to automatically select the number of clusters that are to be used, in such a way that all of the unrelated values would be placed in one cluster, rather then spread out over multiple clusters as seen here in the Distribution of the Query ”0 apr”

Using a search engine to get a basic Normalized ”Bing” Distance as a simple measure of relatedness for the different values worked reasonably well but would need a full Enrichment to Category comparison to be done in order to fully and successfully. With the algorithm averaging out the dataset and clustering the similar distributions led to distributions that removed much of the noise and only kept related and relevant words in the final cluster.

Future Work

This implementation was evaluated with very simple and bare-bones datasets, so there is very little that can be said about it’s success with confidence. To evaluate this algorithm properly, I would have to have a full dataset, with NGD’s for each combination of Word to Category for a decent sized enrichment of each Query.

Based on this I would be able to run the algorithm on each Query and arrive at categorizations that could be compared with the original training data’s categorizations using the precision/recall, used to evaluate categorization algorithms.

Improving the algorithm is also part of this future work. There are still a number of sections that, although they are functional, could be improved to get optimal results from this algorithm. One change that I looked into but never implemented was merging the clusters that had already been created. These clusters could be compared to each other, however they would need to be replaced by another cluster in order for the algorithm to stay at the same number of clusters. This replacement, the selection of which cluster to take a chance at merging and subsequent replacement of the already created cluster adds an element of randomness to the algorithm that had up to this point never been introduced.

Conclusion

In this paper, I have presented a method to categorize a Query given a moderately related corpus of words. The algorithm is successful at selecting most of the categories that were labelled in the training data though a full evaluation would have to be done in order to make conclusions. The results are initially promising and it would be interesting to see how the algorithm would fare against a full enrichment and NGD complete dataset, My thoughts are that it would fare reasonably well. That being said there are still many opportunities to make this a stronger algorithm. There is certainly further room for improvement:

- Evaluation of the project is only partially complete, and though results look promising, only full evaluation would be able to prove success
- Further investigation into the effects of 0 probability categories with the KL algorithm
- Distribution ranking (selection of ”the best”) is simplistic and could be improved
- Selection of the initial seed and merger of existing clusters could result in much more consistent results

I feel that in it’s current state and given the full datasets it would be a highly successful algorithm. The algorithms method of looking at the relationships the enrichment words have with the categories and comparing these relationships as opposed to directly comparing queries with other queries is robust, and returns solid results. While other algorithms showed the effectiveness of distributional clustering on query data, this algorithm shows the potential of distributional clustering on enriched query data.

References

Baker, L. D., and McCallum, A. K. 1998. Distributional clustering of words for text classification. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, 96–103. ACM.

Cilibrasi, R. L., and Vitanyi, P. M. 2007. The google similarity distance. *Knowledge and Data Engineering, IEEE Transactions on* 19(3):370–383.

Hartigan, J. A., and Wong, M. A. 1979. Algorithm as 136: A k-means clustering algorithm. *Applied statistics* 100–108.

Hölscher, C., and Strube, G. 2000. Web search behavior of internet experts and newbies. *Computer networks* 33(1):337–346.

Hu, X.; Zhang, X.; Lu, C.; Park, E. K.; and Zhou, X. 2009. Exploiting wikipedia as external knowledge for document clustering. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, 389–396. ACM.

Kullback, S. 1987. The kullback-leibler distance.

Pereira, F.; Tishby, N.; and Lee, L. 1993. Distributional clustering of english words. In *Proceedings of the 31st annual meeting on Association for Computational Linguistics*, 183–190. Association for Computational Linguistics.

Wen, J.-R.; Nie, J.-Y.; and Zhang, H.-J. 2001. Clustering user queries of a search engine. In *Proceedings of the 10th international conference on World Wide Web*, 162–168. ACM.

Witten, I., and Milne, D. 2008. An effective, low-cost measure of semantic relatedness obtained from wikipedia links. In *Proceeding of AAAI Workshop on Wikipedia and Artificial Intelligence: an Evolving Synergy, AAAI Press, Chicago, USA*, 25–30.