Web Query Categorization
A Non-Straightforward Learning Task

James Bain

May 2016

INTRODUCTION

Web query categorization is a learning task with which the internet-using world is well familiar. Search engines are the starting point for many seeking information retrieval. The information returned by the search engine is expected to be accurate in terms of the user's searching intentions. However, querying the web has inherent problems that make this learning task particularly challenging.

Problems with Web Query Categorization

The way humans query the web poses several challenges for accurately categorizing their searches. Web queries are often very short, between 1 and 4 words long, which make it difficult to determine meaning. Similarly, the words that users choose can also be ambiguous, or hold multiple meanings. For example, if a user queries "Apple store", are they looking for the nearest grocery store that sells apples or are they looking for a local distributor of Apple Inc.'s computer goods? The ability to decipher between these two very reasonable possibilities could mean the return or the departure of a user.

Users can also be naïve to a topic in which they are interested to the extent where the user may make mistakes within their query, such as unintentionally spelling something incorrectly (ie. *Burn, Germany* instead of *Berne, Germany*). The user may incorrectly classify something within the query itself. For example, a user interested in the sign-language using Gorilla, named Koko, may query a search engine for "Koko the monkey", despite the fact that a gorilla isn't actually a monkey, but instead, an ape. Being able to adjust for the ignorant mistakes is an important consideration for a web query categorization task.

The Data

The data for this task was found at `http://calla.rnet.missouri.edu/` and includes three separate files with 800 queries a piece for a total of 2400 unique queries. Each query is a row, and assigned to it are categories that have been designated by a professional. Each query can be assigned up to five separate categories, although having five is not necessary. Each of these categories belong to one of seven super-categories: *Computers*, *Entertainment*, *Information*, *Living*, *Online Community*, *Shopping* and *Sports*. It is these super-categories that I sought to predict. There was no other information provided beyond this making this a non-straightforward learning task. Therefore, features had to be generated before attempting classification.

Feature Generation

Given that the only native input feature to the data was the actual query, I had construct features from the query itself. Past studies interest in query categorization have used the structure of the query to predict category. I focused my efforts on doing something similar. I counted the *number of words per query*, *the number of characters*, and *the number of characters per word*, in hopes that the types of words that users search may influence what they are looking for. I was also interested in seeing whether or not different parts of speech would

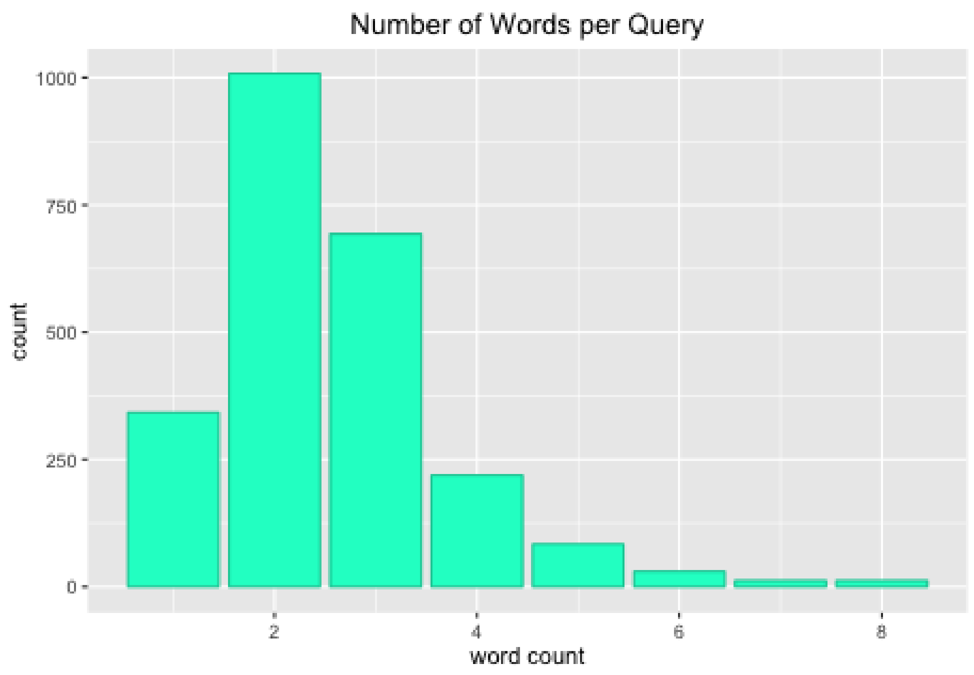

Figure 1: *A bar chart of the number of words per query. The majority of the queries are very short, between 2 and 3 words long. Only a few are above 5 words in length.*

have predictive value for this task. Therefore, the *number of nouns*, *verbs*, and *prepositions* was counted per query. Each of these were then divided by the number of words per query to get the frequency of the parts of speech within the query. Features were all generated in `Python` using the `Pandas`, `Numpy` and `nltk` libraries.

Data Preprocessing

The preprocessing of the data was minimal. However, super-categories had to be extracted from the predefined query categories provided. Each super-category was given its own column in the data set and each query was assigned ”True” if it belonged to that super-category and ”False” otherwise. The configuration was necessary for the machine learning package to perform multi-label classification.

METHODS

The task is to predict the super-categories (these are *Computers*, *Entertainment*, *Information*, *Living*, *Online Community*, *Shopping* and *Sports*) that a query belongs to using the generated features. A query can be categorized as one or more super-categories, making this a multi-label classification task.

Multi-label classification tasks can be handled in one of two ways. The first is to adapt the task into a binary classification problem in which each label is taken as a separate classification task and predicted independently of the other labels. Returned is a prediction of all labels. The second method is to adapt an algorithm to the task to predict all labels

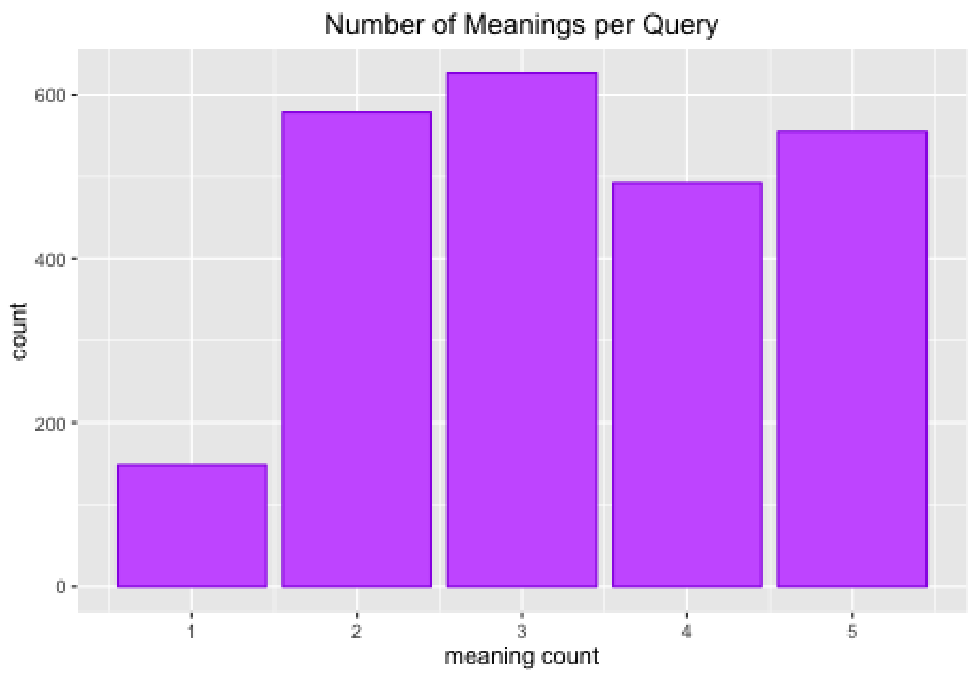

Figure 2: *A bar chart of the number of super-categories assigned per query. Most of the queries contain more than one categorization.*

together, providing more "true-to-the-problem" classification task. Given the programmatic constraints of the latter, I chose to use the binary classification transformation technique for this particular problem.

Algorithms

Using a binary classification wrapper, I compared the performance of a Decision Tree algorithm against that of a Neural Network. Machine learning algorithms were adapted using the `mlr` package in `R`, which takes the learning method, loaded from a dependency package, and accepts it as a parameter within the binary classification wrapper.

Decision Tree

Decision trees are a popular machine learning algorithm that involves traversing down from a root node and making decisions about the classification determined by those feature values similar to the training data labels. Entropy is used to determine the information gain of each feature and decisions are constructed from these features. I trained the data on 2000 of the 2400 rows and tested on the remaining 400 rows of data. The `rpart` package in `R` was used as a parameter in the binary classification wrapper, and each label (super-category) was classified independently of the other.

Neural Network

Neural networks predict labels by find the function given a set of inputs that best describes the label. It does this by randomly initializing a weight, then iterating through and tuning the weight until the error no longer decreases. This function is then used on novel data to find the best label given the inputs. I employed 5 fold cross validation and the performance was assessed on the entire data set. To perform this task, I used the `nnet` package in `R`.

Feature Selection

Many of the features that I generated for this task seemed trivial at first glance. Why would the number of verbs be predictive of category? Therefore, I wanted to calculate the information gain on all of the generated input variables to see if I can reduce the number of inputs going into the task. The information gain of each feature is listed in *Table 1.*

Feature	Information Gain
character count	0.004742764
preposition ratio	0.004949343
noun ratio	0.0
verb ratio	0.0
word count	0.0
char/word	0.007672985

Table 1: *A list of generated input features and the respective information gain associated with each.*

Only those features that provided any information gain were used in the final model. Therefore, the final model consisted of the 7 super-categories being predicted by `character count`, `preposition ratio`, `char/word`, `whether it has a number`, and `the query`.

RESULTS

The performance of the algorithms was assessed using the area under the curve, the precision, the recall, and the F_1 score. Each label was assessed independently as this was a multi-label classification task.

Decision Tree Performance

The Decision Tree for this task did not perform very well. Despite a high rate of correctly classified labels, the area under the curve displays a performance functioning at the same level as random classification for all labels except for `Computers`, which barely performed above what would be expected for random assignment. See *Table 2* for Decision Tree's performance.

Neural Network Performance

The Neural Network, overall, performed better than that of the Decision Tree. The area under the curve was above what would be expected for random classification for all labels, albeit only slightly for `Online Community` and `Living`. In fact, none of the labels had impressively high performance. See *Tabel 3* for the performance of the Neural Network.

Decision Tree								
Label	auc	tp	tn	fp	fn	precision	recall	F_1
Computers	0.548	1	355	1	43	0.5	0.0273	0.0435
Entertainment	0.5	0	294	0	106	NA	0	0
Information	0.5	307	0	93	0	0.766	1	0
Living	0.5	250	0	150	0	0.625	1	0.868
Online Community	0.5	0	286	0	286	NA	0	0.769
Shopping	0.5	0	149	0	149	NA	0	0
Sports	0.5	0	28	0	28	NA	0	0

Table 2: *Performance measures of the Decision Tree for each label.*

Neural Network								
Label	auc	tp	tn	fp	fn	precision	recall	F_1
Computers	0.601	0	2134	0	266	NA	0	0
Entertainment	0.603	13	1755	8	624	0.619	0.020	0.034
Information	0.559	1708	0	692	0	0.712	1	0.832
Living	0.522	1536	1	863	0	0.640	1	0.781
Online Community	0.504	0	1817	0	583	NA	0	0
Shopping	0.546	6	1553	7	834	0.462	0.007	0.014
Sports	0.555	0	2191	0	209	NA	0	0

Table 3: *Performance measures of the Neural Network for each label.*

CONCLUSION

For this non-straightforward web query categorization task, the Neural Network outperformed the Decision Tree. Despite not having exceptionally high predictive capabilities, there appears to be some validity to assessing category based on the structure of the query itself. For example, the characters per word in a query could indicate a vernacular more akin to a certain category than another. Similarly, the use of prepositions could be used more frequently in one category over the others because of the way one would query that topic.

In a very pragmatic sense, however, one would not use these features alone to predict category. Instead, they may serve, more appropriately, as supplemental features to more informative inputs [1]. Data pertaining to the individual querying as well as a time-stamp could be beneficial for such a task, as people tend to query multiple things about one topic in close amount of time.

Future work could seek to generate more latent structural elements of the query. This task would also benefit from more algorithms being tested against the task such as Random Forests as well as approaching the task from the Algorithm Adaptation approach [2] instead of the binary classification transformation.

References

[1] Dou Shen, Rong Pan, Jian-Tao Sun, Jeffrey Junfeng Pan, Kangheng Wu, Jie Yin, and Qiang Yang. Query Enrichment for Web-query Classification. *ACM*, pages 1–33, 2005.

[2] Sheng-Jun Huang, Songcan Chen, and Zhi-Hua Zhou. Multi-Label Active Learning: Query Type Matters. *IJCAI*, pages 946–952, 2005.