Open-Ended Question Classification Using Support Vector Machines

Jim Bullington, Ira Endres, and Muhammad Asadur Rahman

Department of Computer Science
University of West Georgia
Carrollton, GA 30118
{jbullin1, iendres1}@my.westga.edu; mrahman@westga.edu

Abstract

Open-ended questions require a more descriptive response than do closed-ended questions. Questions of this type can be broken down into categories that identify the format and content of the expected response. Accurate classification of open-ended questions can lead to more appropriate responses from question-answering systems and other intelligent systems – leading to better human/computer interaction. This paper demonstrates how support vector machines (SVM) are used successfully in the classification of open-ended questions.

Introduction

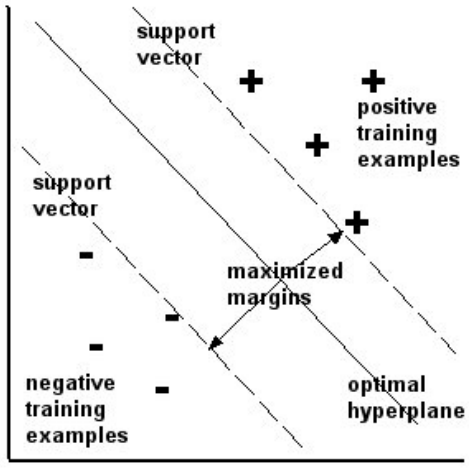
Question-Answering (QA) systems have a goal of returning intelligent responses that are correctly formatted based on the needs of the question – instead of a list of documents, as is the case with Information Retrieval (IR) systems. Accurate classification of questions based on the type of response required is critical to QA systems so that the response is appropriate to the goals set out by the question. In a similar manner, categorization of questions by subject matter has been shown to enhance the accuracy of QA systems by matching a question category to an answer category. In (Moschitti and Harabagiu, 2004), an SVM (along with other techniques) was trained to classify questions into subject categories, which could then be matched to answer categories, thus improving question-answering performance.

For the purpose of this study, we will define an open-ended question as a question that requires a thoughtful answer. Interviewers, marketing people, and instructors use open-ended questions in order to extract as much information as possible from a subject. For example, instead of asking, "Is the number 7 a prime number?" which is a closed-end question requiring only a "yes" or "no" response; the question could be asked, "Explain why the number 7 is a prime number." The latter question requires a descriptive and thoughtful answer.

Open-ended questions can be further categorized into differing classes depending on the question content and wording. Descriptive answers are required for questions such as "Describe the characteristics of an isosceles triangle." A question such as "List the advantages and disadvantages of solar panel based energy generation" calls for a contrasting response. If a technique can be found that is able to accurately classify open-ended questions, this technique could add much value to the problems of human/machine interaction by allowing a system to respond more appropriately to natural language requests.

The Support Vector Machine (SVM) is an emerging machine learning technology that is becoming a more viable alternative to augment and even replace other more mature technologies such as neural networks and nearest neighbor algorithms (Hearst et al., 1998). The Support Vector Machine grew out of research into statistical learning techniques in the late seventies conducted by Vladimir Vapnik and was furthered in the 1990s by Vapnik and others at AT&T Bell Laboratories (Vapnik, 1998). The SVM is a supervised learning machine that is trained by using a dataset that contains feature sets with both positive and negative examples of the data to be categorized. In its simplest form, as shown in Figure 1, the SVM maps the training dataset into a feature space and calculates an optimal hyperplane (or decision boundary) that will separate the positive and negative feature points in such a manner as to maximize the distance (or margin) from each classification boundary (or support vector). (Burges, 1998) Once the training is complete, new feature points can be classified by the SVM by determining where the feature lies in relation to the hyperplane.

Figure 1 – Basic SVM Theory



The open-ended question classification problem is closely related to the text classification problem, in which a machine learner must classify documents that may or may not display a common overlap of keywords or key phrases that establish document class membership. The use of an SVM in text classification has been well researched and found to have good performance without complex natural language processing (Joachims, 2001). The use of an SVM to solve a text classification problem requires that importance weights be assigned to each term in the corpus of training documents - resulting in a highly dimensional feature set. (Leopold and Kindermann, 2002) demonstrates term-to-vector transformations and compares the performance of word-stemming vs. non-stemming along with various weighting factors including: inverse document frequency and redundancy when used as input to an SVM for document retrieval. They show that word stemming along with Term-Frequency-Inverse Document Frequency (TF-IDF) achieves good performance in document recall with a similar SVM as was employed in this research.

It could be argued that perhaps a simple keyword search approach may yield performance equal or better than that of machine learning techniques. However, this would represent a tremendous amount of work and ignores the fact that subtleties may exist in the problem space that make straightforward text parsing and keyword recognition problematic (Zhang and Lee, 2003). Also, the SVM has been shown to have the best classification accuracy in other question classification problems when pitted against other machine learning algorithms (including Nearest Neighbor and Naïve Bayes, for example) (Zhang, and Lee, 2003).

Question Classification and Parsing

The goal of this experiment is to transform the question dataset into feature vectors that can be used to train an SVM to recognize a specific class of open-ended question. In order to accomplish this task, several steps are required. First, a set of open-ended question classes must be established based on the type of answer required for each class. Table 1 lists the classes that will be used to group the training question dataset for this experiment.

For each class of question, training data that contains both positive and negative question examples will be processed and supplied to the SVM. Then the performance of the machine will be tested using other examples that have a known classification. For the purposes of this study, the SVM will be limited to a binary classifier, so each classification will have separate training and testing datasets.

Testing and Training Data

Each question class consists of a set of files containing all questions required for training and testing the SVM in that classification. The set of questions that are to be presented to the SVM are assumed to be in American English and have good sentence syntax, structure, and semantics. No extra language processing is done to ensure that each question makes sense grammatically and intellectually.

Forming Text Attribute Data

In order for the input data to be processed by the SVM, the text must be processed into a word list with attribute vectors. The input dataset for a classification will be filtered to remove symbols and punctuation and to remove stop words such as: a, and, the, etc. Punctuation and stop words have no bearing on the classification of the question. Word stemming will also be applied to reduce words to their root form. Stemming removes inflections and variations from words – such as “s” in plural words, or “ly” in adverbs – so that only the root of the word remains.

Table 1 – Open-Ended Question Classifications

Question Class	Expected Answer Format
Advantage/Disadvantage	Advantages and disadvantages (questions may require certain number)
Cause and Effect	Explain the effect of something on something else
Comparison	Differences/similarities between two or more entities
Definition	Relatively short explanation/description – few lines or few sentences
Example	Explain with an example, provide example
Explanation	More explanation – more descriptive than ‘what’ questions
Identification	Identification of some kind
List	A list of points – may or may not be in sequence
Opinion	Give personal opinion on a particular point or statement – either support or argue
Rationale	Explain why a statement/question is true or false
Significance	Explain the importance of something or why it may be important

This will maximize the term count of the root word and boost the influence of the word in the question class. At this point a word list can be built containing the word stems of the input data. From this word list, a training (and testing) dataset can then be built containing all questions for a classification in the following format:

1. +1 (positive) or –1 (negative) classification
2. pairs of attributes containing the index of a word contained in the question in the word list along with the TF-IDF of the word.

The TF-IDF (Term Frequency – Inverse Document Frequency) is calculated for each term and included as a feature value in the final vector for each example. TF-IDF is frequently used in text processing and document retrieval. The TF-IDF of a term is a weight that indicates the relevance of the term in the corpus of documents. Term Frequency is simply a count of the number of times a term appears in a question. Inverse Document Frequency is a measure of the importance of a term in a set of documents. Thus weight should represent the importance of a term in the question class.

Training the SVM

The resulting vector list is used to train an SVM to correctly distinguish a question that belongs to a particular classification (and one that does not belong). The SVM that is used for this experiment is the SVMLight (Joachims, 1999) program implemented by Thorsten Joachims (this software is available at http://svmlight.joachims.org/). The SVM is executed in training mode and allowed to create a data model for each classification. As discussed previously, the SVM operates by plotting the feature vectors, calculating the support vectors and finding the optimal hyperplane that separates the positive and negative support vectors. Once the SVM is trained, classification can occur by determining where a feature vector lies in relation to the established support vector model.

Testing the SVM

Once the SVM has been trained, testing and evaluation of the SVM model can take place. While executing the SVM software in classification mode, the testing dataset (which has been prepared in the same manner as the training dataset) is provided to the SVM and the classification results are compared against the known (positive or negative) classification of each question. The error rates of the SVM can then be measured and reported.

Experiment Setup

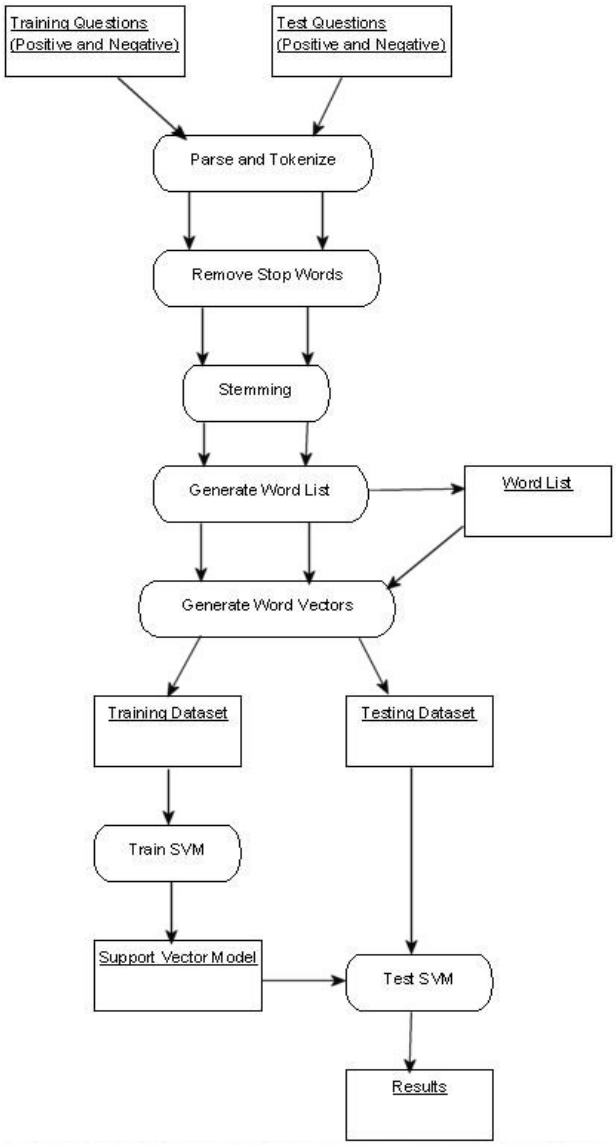
The majority of the processing in this experiment is consumed by the text processing necessary to generate the appropriate input for the SVM. Figure 2 illustrates the steps required to complete the experiment.

As previously discussed, input to the experiment consists of a file of open-ended questions that belong to a certain classification. All of the test data for one classification of question is stored in a single directory with the following file names:

train_pos.txt, train_neg.txt–training examples (pos. and neg.)
test_pos.txt, test_neg.txt –testing examples(pos. and neg.)

In this experiment we used questions from a question bank of 1000 open-ended questions that were created by the authors or inspired from various textbook and reference sources. They covered software engineering and operating systems areas of computer science. Each question was manually segregated into the proper question class and placed into the appropriate training or testing file with no particular ordering observed.

Figure 2 – SVM Experiment Setup



All of the files contain one question per line and each line terminated with a ASCII carriage return / line feed sequence. The files are used to generate an overall word list that can be provided to the word vector generator. In order to generate the most efficient word list, the questions are parsed and tokenized. Punctuation and stop words are removed. Stemming is then applied to the words, so that only the root word of each token remains. The word list is then generated as the final step in this process.

Word vector generation is then performed by comparing each term in each question to the word list in such a way as to generate the required TF-IDF value. The feature vector is then enumerated for each question, noting whether the vector represents a positive (1) or a negative (-1) example. The format of the vector list produced is sparse vectors, where properties (terms) with zero values are not explicitly listed. The index of the term in the word list is specified along with the TF-IDF of the term. An example of the complete text to vector list transformation is given here:

Example Questions from the Explanation Class:

Explain the concept of functions.
Explain the purposes of a device driver.

Word List Generated:

Explain, concept, funct, purpos, devic, driver

Resulting Sparse Vectors (some decimals removed):

1 1:0.208106 2:0.76535539 3:0.60903436
1 1:0.133641 4:0.62165690 5:0.45741152 6:0.62165690

Separate vector list files are produced for training and testing data. The overall number of features in the datasets may reach into the thousands, but the SVMLight implementation is able to handle feature sets of this size without adjustment.

Once the text preparation is complete, the SVM training and testing can proceed. First the SVM is given the training data so that the support vectors can be calculated and the decision boundaries are established. Then, the testing dataset is supplied to the training model so that the SVM can make predictions on the test cases and judge the accuracy of the classification decisions. The SVMLight classification module will log the results of the testing so that the statistics can be analyzed upon completion of the experiment.

Results

The training dataset was given as input to the SVM learner application, then a separate test dataset was given to the SVM classifier application and the resulting data was analyzed. The results of the experiment are summarized in Table 2. The training dataset consisted of equal numbers of both positive and negative examples of each question classification. The testing dataset contained both positive and negative examples as well, so the SVM was judged on the precision and accuracy of classifying both positive and negative examples of each classification. The magnitudes of the term weights were normalized to [0, 1] which improves the accuracy of the SVM. The feature vectors were kept in the same order as the question lists, so that incorrect identifications could be easily traced back to the original question. SVMLight provides several log files that are invaluable for troubleshooting the classifier and determining the final performance statistics.

Overall, the SVM classifier was, on average, 74.6% accurate in classifying both positive and negative test examples for each class of open-ended question. The classifier displayed the greatest accuracy with the Significance class (92.9%); the least accuracy with the Identification class (50.0%).

Table 2 – SVM Classification Results

Question Classification	Total Training Examples (Pos. & Neg.)	Accuracy	Precision/ Recall
Advantage/ Disadvantage	30	80.0%	66.7% / 80.0%
Cause and Effect	30	73.3%	55.6% / 100.0%
Comparison	100	87.5%	85.7% / 90.0%
Definition	240	68.3%	64.9% / 80.0%
Example	60	55.6%	50.0% / 50.0%
Explanation	300	83.3%	83.3% / 50.0%
Identification	60	50.0%	66.7% / 37.5%
List	30	86.7%	80.0% / 80.0%
Opinion	60	75.0%	72.7% / 80.0%
Rationale	60	70.0%	70.0%/ 70.0%
Significance	30	92.9%	100.0% / 80.0%

Precision measures the percentage of positive examples that were correctly identified as opposed to the total number of correct and incorrect positive examples identified and is given by:

$$\text{Precision} = \frac{TruePositives}{TruePositives + FalsePositives}$$

The Significance class achieved the highest precision (100%) and the Example class had the lowest precision (50.0%).

Recall (also known as sensitivity) measures the percentage of positive examples that were correctly identified as opposed to the total number of positive examples that existed and is given by:

$$\mathrm{Re}\,call = \frac{TruePositives}{TruePositives + FalseNegatives}$$

The Cause and Effect class attained the highest recall (100.0%) and the Identification class had the lowest recall (37.5%).

Conclusions

In training the SVM for this task, we were hoping that the SVM could be trained to recognize the occurrence of certain keywords or phrases in a question class and then, based on the recurrence of these same keywords, be able to correctly identify a question as belonging to that class. In the same way, if a question did not contain certain keywords, the SVM should distinguish the question as not belonging to the question class being tested. In our trials, we had mixed results - with some classes performing better than others.

In analyzing the test results, there does not seem to be a correlation between the number of training examples and the accuracy of the classifier. For even though the Explanation class contained the most training examples (300), the SVM did not exhibit the most accuracy in this class. In fact, the most accurate class (Significance – 92.9%) was among the classes with the fewest examples.

The poor results of the SVM classifier may be the result of ambiguous terms appearing in some of the classifications. For example, in the Identification class the keyword "what" appears in most of the positive training and testing examples. This same keyword is also a common word in other question classes as well, which may cause some ambiguity in the classifier. The low recall percentage of the Identification class shows that the classifier associated more of the features in the positive test examples with the negative training examples than it associated with the positive training examples.

Better results were obtained in classes that contained keywords that were uncommon in the other categories. For instance the Significance class contains the keywords "significance" and "importance" which are uncommon in the other classes. This is also the case in the Comparison class, which contains words such as "compare" or "differences" which are also infrequent in the other classes.

We can conclude that a properly trained Support Vector Machine is able to accurately classify open-ended questions. But occurrences of common keywords that frequently appear across question classifications results in lowered accuracy.

References

Burges, C. J. C. (1998). "A Tutorial on Support Vector Machines for Pattern Recognition." *Data Mining and Knowledge Discovery*, 2:121–167.

Hearst, M. A., Schölkopf B., Dumais S., Osuna E., and Platt J. (1998). "Trends and Controversies-Support Vector Machines." *IEEE Intelligent Systems*, 13(4): 18-28.

Joachims, T. (1999). "Making Large-Scale SVM Learning Practical." In T. Joachims, *Advances in Kernel Methods - Support Vector Learning.* Cambridge, MA: MIT Press.

Joachims, T. (2001)."A Statistical Learning Model of Text Classification With Support Vector Machines." In *Proceedings of SIGIR. 2001*, NY:ACM Press,128–136.

Leopold, E., and Kindermann, J. (2002). "Text Categorization With Support Vector Machines: How to Represent Text in Input Space?" *Machine Learning*, 46(3):423–444.

Moschitti, A. and Harabagiu, S. (2004). "A Novel Approach to Focus Identification to Question/Answer Systems." *Proceedings of the Workshop on Pragmatics of Question Answering at HLT-NAACL 2004*, 43-51.

Vapnik, V. *Statistical Learning Theory*. John Wiley and Sons, Inc. New York. 1998.

Zhang, D., Lee, W. S. (2003) "Question classification using support vector machines." *Proceedings of SIGIR 2003*, NY:ACM Press, 26-32.