Simple or Complex?
Classifying the Question by the Answer Complexity

Yllias Chali *Sadid A. Hasan*
University of Lethbridge, Lethbridge, AB, Canada
chali@cs.uleth.ca, hasan@cs.uleth.ca

ABSTRACT
Simple questions require small snippets of text as the answers whereas *complex* questions require inferencing and synthesizing information from multiple documents to have multiple sentences as the answers. The traditional QA systems can handle simple questions easily but complex questions often need more sophisticated treatment e.g. question decomposition. Therefore, it is necessary to automatically classify an input question as simple or complex to treat them accordingly. We apply two machine learning techniques and a Latent Semantic Analysis (LSA) based method to automatically classify the questions as simple or complex.

KEYWORDS: Simple questions, complex questions, support vector machines, k-means clustering, latent semantic analysis.

Proceedings of the Workshop on Question Answering for Complex Domains, pages 1–10,
COLING 2012, Mumbai, December 2012.

1 Introduction

Automated Question Answering (QA), the ability of a machine to answer questions asked in natural language, is perhaps the most exciting technological development of the past few years (Strzalkowski and Harabagiu, 2008). QA research attempts to deal with a wide range of question types including: fact, list, definition, how, why, hypothetical, semantically-constrained, and cross-lingual questions. This paper concerns open-domain question answering where the QA system must handle questions of different types: simple or complex.

Simple questions are easier to answer (Moldovan et al., 2007) as they require small snippets of texts as the answers. For example, with a simple (i.e. factoid) question like: "What is the magnitude of the earthquake in Japan?", it can be safely assumed that the submitter of the question is looking for a number. Current QA systems have been significantly advanced in demonstrating finer abilities to answer simple factoid and list questions. On the other hand, with complex questions like: "How is Japan affected by the earthquake?", the wider focus of this question suggests that the submitter may not have a single or well-defined information need. Therefore, to answer complex type of questions we often need to go through complex procedures such as question decomposition and multi-document summarization (Chali et al., 2012; Harabagiu et al., 2006; Chali and Hasan, 2012; Chali et al., 2009). Hence, it is necessary to automatically classify an input question as simple or complex in order to answer them using the appropriate technique. Once we classify the questions as simple or complex, we can pass the simple questions to the traditional question answering systems whereas complex questions can be tackled differently in a sophisticated manner. For example, the above complex question can be decomposed into a series of simple questions such as *"How many people had died by the earthquake?"*, *"How many people became homeless?"*, and *"Which cities were mostly damaged?"*. These simple questions can then be passed to the state-of-the-art QA systems, and a single answer to the complex question can be formed by combining the individual answers to the simple questions (Harabagiu et al., 2006; Hickl et al., 2006). This motivates the significance of classifying a question as simple or complex. We experiment with two well-known machine learning methods and show that the task can be accomplished effectively using a simple feature set. We also use a LSA-based technique to automatically classify the questions as simple or complex.

2 Question Classification

Question classification is the task of assigning class labels to a given question posed in natural language. The main objective of question classification is to deal with a group of similar questions in a similar fashion, rather than focusing on each question individually. Researchers have shown that the performance of a QA system could further improve if question classification is employed (Ittycheriah et al., 2001; Hovy et al., 2001; Moldovan et al., 2003). Most approaches to question classification are based on complex natural language processing techniques which extract useful information from the question and utilize that to answer the question in an effective manner. Different rule-based and learning-based techniques have been applied over the years to tackle the question classification task (Prager et al., 1999; Silva et al., 2011; Bu et al., 2010; Zhang and Lee, 2003; Moschitti and Basili, 2006; Li and Roth, 2002).

In order to classify the questions as simple or complex we experiment with two machine learning techniques: 1) supervised and 2) unsupervised. Supervised classifiers are typically trained on data pairs, defined by feature vectors and corresponding class labels. On the other hand, unsupervised approaches rely on heuristic rules and work on unlabeled data. In this paper, we

employ SVM for supervised learning whereas for the unsupervised learning experiment we use k-means clustering algorithm. We also accomplish the task using a LSA-based methodology where the main idea is to exploit a training corpus of already classified questions and then, to compare the test set questions with the semantic space of the training corpus to identify their class.

2.1 SVM

SVM is a powerful methodology for solving machine learning problems introduced by Vapnik (Cortes and Vapnik, 1995) based on the Structural Risk Minimization principle. In the field of natural language processing, SVMs are applied to text categorization and syntactic dependency structure analysis, and are reported to have achieved higher accuracy than previous approaches (Joachims, 1998). SVMs were also successfully applied to part–of–speech tagging (Giménez and Màrquez, 2003), single document summarization for both Japanese (Hirao et al., 2002a) and English documents (Hirao et al., 2002b), and multi-document summarization (Chali and Hasan, 2012; Hirao et al., 2003; Schilder and Kondadadi, 2008). This motivates us to employ SVM in our task. In the classification problem, the SVM classifier typically follows from the solution to a quadratic problem. SVM finds the separating hyperplane that has maximum margin between the two classes in case of binary classification. SVMs can also handle non-linear decision surfaces introducing kernel functions (Joachims, 1998; Kudo and Matsumoto, 2001). We consider our problem as binary classification having two classes: 1) simple questions and 2) complex questions.

In SVM, the training samples each of which belongs either to positive or negative class can be denoted by:

$$\left(x_1, y_1\right), \ldots, \left(x_u, y_u\right),\ x_j \in R^n,\ y_j \in \{+1, -1\}$$

Here, x_j is a feature vector of the j-th sample represented by an n dimensional vector; y_j is its class label. u is the number of the given training samples. SVM separates positive and negative examples by a hyperplane defined by:

$$w \cdot x + b = 0,\ w \in R^n, b \in R \tag{1}$$

Where "·" stands for the inner product. In general, a hyperplane is not unique (Cortes and Vapnik, 1995). The SVM determines the optimal hyperplane by maximizing the margin. The margin is the distance between negative examples and positive examples; the distance between $w \cdot x + b = 1$ and $w \cdot x + b = -1$. The examples on $w \cdot x + b = \pm 1$ are called the Support Vectors which represent both positive or negative examples. The hyperplane must satisfy the following constraints:

$$y_i\left(w \cdot x_j + b\right) - 1 \geq 0$$

Hence, the size of the margin is $2/||w||$. In order to maximize the margin, we assume the following objective function:

$$\begin{aligned} Minimize_{w,b} J(w) &= \frac{1}{2}||w||^2 \\ s.t.\ y_j\Big(w \cdot x_j + b\Big) - 1 &\geq 0 \end{aligned} \tag{2}$$

By solving a quadratic programming problem, the decision function $f(x) = sgn\Big(g(x)\Big)$ is derived, where

$$g(x) = \sum_{i=1}^{u} \lambda_i y_i x_i \cdot x + b \tag{3}$$

SVMs can handle non-linear decision surfaces with kernel function $K\left(x_i \cdot x\right)$. Therefore, the decision function can be rewritten as follows:

$$g(x) = \sum_{i=1}^{u} \lambda_i y_i K\left(x_i, x\right) + b \tag{4}$$

In this research, we use the linear kernel functions, which have been found to be very effective in the study of other tasks in natural language processing (Joachims, 1998; Kudo and Matsumoto, 2001).

2.2 k-means Clustering

In cluster analysis, the data or samples are divided into a number of useful subsets based on the similarity of data points. Initially, the number of subsets (clusters) or how they are distinguished from each other is not known since the training data are not labeled with the class information. k-means is a hard clustering algorithm that defines the clusters by the center of mass of their members (Manning and Schutze, 2000). It starts with a set of initial cluster centers and goes through several iterations of assigning each data object (i.e. each question in our case) to the cluster whose center is the closest. The k-means algorithm follows a simple way to cluster a given data set through a pre-specified number of clusters k. In our task, we simply assume the number of clusters, $k = 2$ since we have two clusters of questions: 1) simple and 2) complex. After all objects have been assigned, we recompute the center of each cluster as the centroid or mean ($\boldsymbol{\mu}$) of its members. We use the squared Euclidean distance as the distance function. Once we have learned the means of the clusters using the k-means algorithm, our next task is to rank the sentences according to a probability model. We have used Bayesian model in order to do so:

$$\begin{aligned} P(q_k|\boldsymbol{x}, \Theta) &= \frac{p(\boldsymbol{x}|q_k, \Theta)P(q_k|\Theta)}{p(\boldsymbol{x}|\Theta)} \\ &= \frac{p(\boldsymbol{x}|q_k, \Theta)P(q_k|\Theta)}{\sum_{k=1}^{K} p(\boldsymbol{x}|q_k, \Theta)p(q_k|\Theta)} \end{aligned} \tag{5}$$

where q_k is a cluster, $\mathbf{x}$ is a feature vector representing a sentence and Θ is the parameter set of all class models. We set the weights of the clusters as equiprobable (i.e. $P(q_k|\Theta) = 1/K$). We calculated $p(\boldsymbol{x}|q_k, \Theta)$ using the Gaussian probability distribution. The Gaussian probability density function (pdf) for the d-dimensional random variable $\boldsymbol{x}$ is given by:

$$p_{(\boldsymbol{\mu},\boldsymbol{\Sigma})}(\boldsymbol{x}) = \frac{e^{\frac{-1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})}}{\sqrt{2\pi}^d\sqrt{det(\boldsymbol{\Sigma})}} \tag{6}$$

where $\boldsymbol{\mu}$, the mean vector and $\boldsymbol{\Sigma}$, the covariance matrix are the parameters of the Gaussian distribution. We get the means ($\boldsymbol{\mu}$) from the k-means algorithm and we calculate the covariance matrix using the unbiased covariance estimation procedure:

$$\hat{\boldsymbol{\Sigma}_j} = \frac{1}{N-1}\sum_{i=1}^{N}(\boldsymbol{x}_i - \boldsymbol{\mu}_j)(\boldsymbol{x}_i - \boldsymbol{\mu}_j)^T \tag{7}$$

2.3 LSA

LSA (Landauer et al., 1998) uses a sophisticated approach to decode the inherent relationships between the contexts (typically a sentence, a paragraph or a document) and the words that they contain. The main ability of LSA is to identify the similarity between two texts even they do not have any words in common, thus providing at least a similarity score by taking synonymy and polysemy into consideration. In the first phase of LSA, a word-by-context (WCM) matrix is constructed that represents the number of times each distinct word appears in each context. The next phase is called the dimensionality reduction step. In this phase, the dimension of the WCM is shortened by applying Singular Value Decomposition (SVD) and then reducing the number of singular values in SVD. This is done in order to access the ability of LSA in determining similarity scores (other than zero) in case where two documents have nothing in common between them. To accomplish our classification task, we prepare a training corpus of two documents that contain already classified simple and complex questions. Then, the test questions are compared with the semantic space of this corpus and the question that has the highest similarity score to a document is placed under that class. For example, if a test question shows a higher similarity score with the semantic space of the document containing simple questions, it is labeled as a simple question.

3 Experiments

3.1 Data Preparation

The well-known data set[1] available for question classification is created by Li and Roth (2002). For our experiments, we have used a modified version of this data set. The original data set consists of $5,452$ annotated questions for training. All these questions are labeled (i.e. annotated) as one of the six coarse-grained categories: ABBR (e.g. What does NAFTA stand

[1] http://cogcomp.cs.illinois.edu/Data/QA/QC/

for?), DESC (e.g. Why did the world enter a global depression in 1929?), ENTY (e.g. What color are tennis balls?), HUM (e.g. What is Nicholas Cage 's occupation?), LOC (e.g. What province is Edmonton located in?) and NUM (e.g. How many lawyers are there in the state of New Jersey?). An extensive manual analysis of this data set reveals the fact that DESC type questions need complex processing whereas all other types of questions are simple questions that can be answered by the QA systems easily. From the original data, we extract the 5,452 training questions[2] and then, assign new labels to them ($+1$ for simple questions and -1 for complex questions). The 2005, 2006 and 2007 Document Understanding Conferences (DUC[3]) focus on the task of complex question answering. They provide a list of topics along with topic descriptions (having complex questions) and a collection of relevant documents (that contains the required answers). We collect the complex questions from the topic descriptions of DUC-2006 and DUC-2007 and mix them with the previously labeled data set[4] (after assigning the label -1). Thus, we produce a labeled data set of $5,542$ questions where 4144 of them are simple questions and 1398 are complex questions.

3.2 Feature Space

For the machine learning experiments, we represent each question as a vector of feature-values. We extract the following boolean features automatically from the questions. In presence of a certain feature, we set the corresponding feature-value to 1 or assign 0, otherwise. In addition to these, we also consider the length (i.e. number of words) of a question as a useful feature. All the feature-values are normalized to [0, 1] at the end.

3.2.1 First Unigram (Which, Where, Who, What, When)

Simple questions mostly start with the unigram: *which, where, who, what or when*. We assign the value 1 if any of these five question words is present as the first unigram in the question.

3.2.2 Imperative Sentences as Questions

Some complex questions in DUC-2006 and DUC-2007 were formed as imperative sentences. These questions give instructions or express requests for some information or a particular answer (e.g. "Describe developments in the movement for the independence of Quebec from Canada."). If a question is an imperative sentence, we give it the score 1. At the same time, we also look for the question word *how* as the first unigram since a good number of complex questions begin with *how* (e.g. How do you write a book report?).

3.2.3 First Bigram (Starting with How)

The first bigram of several simple questions can be any of the following: *how many, how much, how long, how large, how big, how fast, how small* etc. We look for the presence of this type of bigrams and set the feature-value to 1, if found.

3.2.4 First Bigram (Starting with Who, What)

The bigrams: *Who is, who are, what is, what are* are often found in both simple and complex type of questions. We set the feature-value to 1 if this type of bigram is present in a question.

[2]Only questions are extracted (i.e. without their coarse-grained labels).
[3]http://duc.nist.gov/
[4]DUC complex questions are added into the dataset in order to include variety in the question space.

3.3 System Settings

For the SVM experiments, we use SVM^{light} package[5]. To allow some flexibility in separating the classes, SVM models have a cost parameter, C. We keep the value of C as default and use the linear kernel to run our experiments. For the k-means experiments, we use the k-means implementation[6] of (Pelleg and Moore, 1999). We keep the initial number of centers to 1 and use the default values of other parameters. For the LSA experiment, we use a publicly available implementation[7]. A stopword list is used to exclude unnecessary words from the WCM construction. We delete question words from the stopword list since question words are important for our task. We do not apply dimension reduction in LSA as this setting gives us the most accurate scores[8].

3.4 Evaluation and Analysis

Our data set consists of 5,542 annotated questions. We split the data set into three equal portions to apply 3-fold cross validation for the SVM and LSA experiments. In run-1, we use the first two portions as training data and the last portion as validation (i.e. testing) data. Similarly, in run-2 and run-3, we use the first and the third subset of data for testing, respectively. On the other hand, after a number of iterations (maximum 200), the k-means algorithm converges and each question is assigned to the cluster whose center is the closest according to the Euclidean distance function. We form three different data sets for the k-means experiments. In run-1, we use 1,848 questions for learning while in run-2 and in run-3, we use 3,695 and 5,542 questions, respectively. We can judge the performance of a classifier by calculating its *accuracy* on a particular test set. The accuracy can be defined as:

$$Accuracy = \frac{no.\ of\ Correctly\ Classified\ Questions}{Total\ no.\ of\ Test\ Questions}$$

In Table 1 to Table 3, we show the results for our SVM, k-means and LSA experiments. From the results we can see that the unsupervised k-means classifier clearly outperforms the supervised SVM classifier and the LSA-method for the considered task. This is due to the fact that for supervised learning and LSA experiments we need a huge number of labeled data for training. And also, the training set should be balanced having an equal distribution of the class samples. Our data set had a comparatively less number of complex questions than the simple questions. This might be the reason why SVM and LSA showed a lower accuracy than the k-means classifier. However, the average accuracy of SVM is still near 80.00% showing its good generalization ability. On the other hand, k-means classifier shows the average accuracy of 93.33% that yields the fact that it could learn from the given data set quite remarkably. This phenomenon also suggests that the k-means classifier could learn well from a skewed distribution of simple and complex questions, and this high performance is not due to overfitting on the data. Besides, the lower average accuracy of LSA suggests that the semantic understanding of the questions' content was not 100% accurate. We conduct a similar experiment with a sample dataset of 2796 questions having uniform distribution of simple and complex questions and find that SVM,

[5] http://svmlight.joachims.org/

[6] http://www.cs.cmu.edu/ dpelleg/kmeans/

[7] http://code.google.com/p/lsa-lda/

[8] We experimented with different dimensions while creating the semantic space with LSA, but, dimension reductions produced lesser accuracy in results.

k-means and LSA show an average accuracy of 83.18%, 81.64%, and 70.90%, respectively. From these results, we can see that the supervised SVM system outperforms the unsupervised k-means system when there is a uniform distribution of the question types. We can also see that the LSA system is showing a higher accuracy, which justifies the effectiveness of the approach.

Experiment	Accuracy (in %)
Run-1	79.97%
Run-2	78.94%
Run-3	79.65%
Average	79.52%

Table 1: Accuracy of SVM

Experiment	Learning Data Size	Accuracy (in %)
Run-1	1848	93.45%
Run-2	3695	93.50%
Run-3	5542	93.05%
Average	–	93.33%

Table 2: Accuracy of k-means

Experiment	Accuracy (in %)
Run-1	60.20%
Run-2	62.28%
Run-3	61.33%
Average	61.27%

Table 3: Accuracy of LSA

Conclusion

We perform the task of automatically classifying questions (that are given as input to a standard QA system) as simple or complex. This task is important because it can help a QA system decide what particular actions are needed to be taken to treat the simple or complex questions differently in an effective manner. We use two machine learning techniques: a) supervised SVM and b) unsupervised k-means algorithm, and show that the task can be accomplished effectively using a simple feature set. We also use a LSA-based technique to automatically classify the questions as simple or complex. Extensive experiments show the effectiveness of our proposed approach. In future, we plan to use more sophisticated features and then, experiment with other machine learning techniques on a larger data set.

Acknowledgments

The research reported in this paper was supported by the Natural Sciences and Engineering Research Council (NSERC) of Canada – discovery grant and the University of Lethbridge. The authors gratefully acknowledge the assistance provided by Siddharth Subramanian.

References

Bu, F., Zhu, X., Hao, Y., and Zhu, X. (2010). Function-based question classification for general qa. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, EMNLP '10, pages 1119–1128. ACL.

Chali, Y. and Hasan, S. A. (2012). Query-focused Multi-document Summarization: Automatic Data Annotations and Supervised Learning Approaches. *Journal of Natural Language Engineering*, 18(1):109–145.

Chali, Y., Hasan, S. A., and Imam, K. (2012). Learning Good Decompositions of Complex Questions. In *Proceedings of the 17th International Conference on Applications of Natural Language Processing to Information Systems (NLDB 2012)*, pages 104–115. Springer-Verlag.

Chali, Y., Joty, S. R., and Hasan, S. A. (2009). Complex Question Answering: Unsupervised Learning Approaches and Experiments. *Journal of Artificial Intelligence Research*, 35:1–47.

Cortes, C. and Vapnik, V. N. (1995). Support Vector Networks. *Machine Learning*, 20:273–297.

Giménez, J. and Màrquez, L. (2003). Fast and accurate part-of-speech tagging: The svm approach revisited. In *RANLP*, pages 153–163.

Harabagiu, S., Lacatusu, F., and Hickl, A. (2006). Answering complex questions with random walk models. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 220 – 227. ACM.

Hickl, A., Wang, P., Lehmann, J., and Harabagiu, S. (2006). Ferret: Interactive question-answering for real-world environments. In *Proceedings of the COLING/ACL on Interactive presentation sessions*, pages 25–28.

Hirao, T., Isozaki, H., Maeda, E., and Matsumoto, Y. (2002a). Extracting Important Sentences with Support Vector Machines. In *Proceedings of the 19th International Conference on Computational Linguistics*, pages 1–7, Taipei, Taiwan.

Hirao, T., Sasaki, Y., Isozaki, H., and Maeda, E. (2002b). NTT's Text Summarization System for DUC 2002. In *Proceedings of the Document Understanding Conference*, pages 104–107, Philadelphia, Pennsylvania, USA.

Hirao, T., Suzuki, J., Isozaki, H., and Maeda, E. (2003). NTT's Multiple Document Summarization System for DUC 2003. In *Proceedings of the Document Understanding Conference*, Edmonton, Canada.

Hovy, E., Gerber, L., Hermjakob, U., Lin, C. Y., and Ravichandran, D. (2001). Toward semantics-based answer pinpointing. In *Proceedings of the first international conference on Human language technology research*, pages 1–7.

Ittycheriah, A., Franz, M., Zhu, W. J., Ratnaparkhi, A., and Mammone, R. J. (2001). IBM's statistical question answering system. In *Proceedings of the 9th Text Retrieval Conference*.

Joachims, T. (1998). Text Categorization with Support Vector Machines: Learning with Many Relevant Features. In *European Conference on Machine Learning (ECML)*.

Kudo, T. and Matsumoto, Y. (2001). Chunking with Support Vector Machine. In *Proceedings of the 2nd Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL 2001)*, pages 192–199, Carnegie Mellon University, Pittsburgh, PA, USA.

Landauer, T. K., Foltz, P. W., and Laham, D. (1998). An Introduction to Latent Semantic Analysis. *Discourse Processes*, (25):259–284.

Li, X. and Roth, D. (2002). Learning Question Classifiers. In *Proceedings of the 19th International Conference on Computational Linguistics*, pages 1–7, Taipei,Taiwan.

Manning, C. D. and Schutze, H. (2000). *Foundations of Statistical Natural Language Processing*. The MIT Press.

Moldovan, D., Clark, C., and Bowden, M. (2007). Lymba's PowerAnswer 4 in TREC 2007. In *TREC*.

Moldovan, D., Paşca, M., Harabagiu, S., and Surdeanu, M. (2003). Performance issues and error analysis in an open-domain question answering system. *ACM Trans. Inf. Syst.*, 21:133–154.

Moschitti, A. and Basili, R. (2006). A Tree Kernel Approach to Question and Answer Classification in Question Answering Systems. In *Proceedings of the 5th International Conference on Language Resources and Evaluation*, Genoa, Italy.

Pelleg, D. and Moore, A. (1999). Accelerating exact k-means algorithms with geometric reasoning. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 277–281. ACM.

Prager, J., Radev, D., Brown, E., and Coden, A. (1999). The Use of Predictive Annotation for Question Answering in TREC8. In *In NIST Special Publication 500-246:The Eighth Text REtrieval Conference (TREC 8*, pages 399–411. NIST.

Schilder, F. and Kondadadi, R. (2008). FastSum: Fast and Accurate Query-based Multi-document Summarization. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies: Short Papers*, pages 205–208. ACL.

Silva, J., Coheur, L., Mendes, A. C., and Wichert, A. (2011). From symbolic to sub-symbolic information in question classification. *Artificial Intelligence Review*, 35:137–154.

Strzalkowski, T. and Harabagiu, S. (2008). *Advances in Open Domain Question Answering*. Springer.

Zhang, A. and Lee, W. (2003). Question Classification using Support Vector Machines. In *Proceedings of the Special Interest Group on Information Retrieval*, pages 26–32, Toronto, Canada. ACM.