Question Classification Using Extreme Learning Machine on Semantic Features

Hardy[1] & Yu-N Cheah[2]

[1]Department of Computer Science, STMIK Mikroskil,
Jalan Thamrin No. 140, Medan 20212, Indonesia
[2]School of Computer Sciences, Universiti Sains Malaysia,
11800 USM Penang, Malaysia
Email: hardy@mikroskil.ac.id

Abstract. In statistical machine learning approaches for question classification, efforts based on lexical feature space require high computation power and complex data structures. This is due to the large number of unique words (or high dimensionality). Choosing semantic features instead could significantly reduce the dimensionality of the feature space. This article describes the use of Extreme Learning Machine (ELM) for question classification based on semantic features to improve both the training and testing speeds compared to the benchmark Support Vector Machine (SVM) classifier. Improvements have also been made to the head word extraction and word sense disambiguation processes. These have resulted in a higher accuracy (an increase of 0.2%) for the classification of coarse classes compared to the benchmark. For the fine classes, however, there is a 1.0% decrease in accuracy but is compensated by a significant increase in speed (92.1% on average).

Keywords: *extreme learning machine; head word feature; question classification; semantic features; statistical question classification.*

1 Introduction

The success of sentence retrieval and answer extraction efforts rely on the identification of two important feedbacks: the answer type and the answer context [1]. The answer type determines what kind of answer the question expects, while the answer context determines in what context the answer appears in. Determining the right answer type and answer context can be achieved through **Question Classification** (QC).

Currently, the state-of-the-art machine learning approach in QC is Huang, *et al.* [2]'s approach with the Support Vector Machine (SVM) [3] classifier. However, a newer feed forward neural network method such as Extreme Learning Machine (ELM) [4] that uses fewer nodes than SVM provides faster learning performance than SVM without sacrificing accuracy. These properties are shown in Huang, *et al.* [4]'s experiments on many different cases such as question classification, diabetes detection, and forest cover type prediction.

Received February 7th, 2013, Revised July 5th, 2013, Accepted for publication July 5th, 2013.

Huang, *et al*. [4] also reported that the ELM performed better in terms of accuracy and speed compared to SVM and Backpropagation [5].

There are two approaches in QC: statistical and non-statistical. The statistical approach predicts the question class based on patterns that are found after statistically analyzing the question sentences. The statistical approach is typically performed using machine learning. Non-statistical approaches, on the other hand, uses hand-crafted rules that are formulated based on question and answer structures to predict the question class. Although hand-crafted rules can be very accurate in predicting certain types of question classes, it is difficult to cover a large number of question syntactical structures.

The statistical approach to QC entails the following steps: building a feature space from the training data, learning the patterns from the feature space, and predicting the class of testing dataset. The feature space from the first step may consist of several possible types of features. Commonly used features are: lexical (e.g. bag-of-words), syntactic and semantic. Lexical feature are well-known to have high-dimensional feature space due to the large number of unique possible words. Classifying on a high-dimensional feature space requires high computation power and complex data structures. Alternatively, choosing semantic features instead could significantly reduce the dimensionality of the feature space and increase the accuracy of question classification if they are properly extracted from the training dataset [2].

The main contribution of this research is the introduction of ELM as a new classifier for statistical question classification. Other contributions include the improvements to Huang, *et al*. [2]'s algorithm for head word extraction such as the addition of more regular expressions and modification of noun phrase extraction; replacement of the existing word sense disambiguation (WSD) method with a newer approach; and modification of Huang, *et al*. [2]'s implementation of Collins Head Finder rules [6]. The improvements in the feature selection process have resulted in a higher accuracy (an increase of 0.2%) for the classification of coarse classes compared to Huang, *et al*. [2]'s results. For the fine classes, however, there is a 1.0% decrease in accuracy but is compensated by a significant increase in speed (92.1% on average). Our aim is to improve the accuracy and speed of existing question classification approaches.

2 Related Work

The process of QC involves parsing the training set of question sentences into a feature space, extracting patterns from the feature space, and classifying question sentences from the testing dataset based on those extracted patterns. In this section, we distinguish between the non-statistical and statistical approaches for question classification, as well as present details on classifier features and classifier engine.

2.1 Non-Statistical Classification

Non-statistical classification uses hand-crafted rules to identify question classes. One such effort is by Pasca & Harabagiu [7]. These rules are very efficient. However, they are not practical in identifying questions with various sentence structures. It was also very time consuming to make each rule for every possible type of question.

Rules were also used in Silva, *et al*. [8]'s efforts through direct matching for specific questions as well as by identifying head words that are then mapped into the question classification by using WordNet [9]. The rule-based question classifier was then enhanced using a SVM resulting in an improvement in classification compared to the stand-alone rule-based classifier.

2.2 Statistical Classification

In statistical classification, patterns are extracted automatically from the question sentence training set instead of using predefined rules. This solved problems of non-statistical classification such as difficulty in handling similar questions with different words and syntactic structures using a small set of rules. Several statistical classification approaches will be discussed further.

2.2.1 Li & Roth's Question Classifier

Li & Roth [10] developed a question classifier that comprises two sequential classifiers [11]. These two classifiers were based on Sparse Network of Winnows (SNoW) algorithm [12] which is a multi-class classifier that is specific to high-dimensional feature space classification. Li & Roth [10] reported that the classifier's best result was 78.8% for the fine classes.

Features that were used could be divided into two types: syntactic and lexical semantic. The part-of-speech (POS) tag, chunk and head chunk are of the syntactic type. Named entity, semantically related word, and relational feature [13] are of the lexical semantic type.

2.2.2 Hacioglu & Ward's Question Classifier

Hacioglu & Ward [14] used SVM as their classifier. They modified the multi-class classification SVM into a binary classification SVM by applying the error correcting output coding (ECOC) [15] method.

For feature space, bag-of-words, n-gram, and named entity [16] were used. Dimensionality was reduced by applying the singular value decomposition

(SVD) method, although it was also reported that SVD lowered the accuracy. The best result of the classifier was 82% for the fine classes.

2.2.3 Zhang & Lee's Question Classifier

Zhang & Lee [17] used several machine learning techniques: SVM with tree kernel, nearest neighbour (NN), naive Bayes (NB), decision tree (DT), and SNoW. Bag-of-words, n-gram and tree kernel features were used to represent the feature space.

Tree kernel was proposed as an improvement to the existing SVM approach, allowing better computation in a high-dimensional feature space. The best results obtained were 90% for the coarse classes and 79.2% for the fine classes. The best results for the fine class classification, however, were actually obtained by using linear SVM and not tree kernel SVM.

2.2.4 Krishnan, *et al.*'s Question Classifier

In Krishnan, *et al.* [18]'s work, a SVM similar to that of Zhang & Lee [17]'s work was employed. For the feature set, bag-of-words, n-gram, answer type informer span and its hypernyms were used. The informer span serves as a clue for the correct identification of a question class. However, for the hypernym feature, WSD was not employed; all the possible senses of a given word were inserted into the feature space instead. The best results obtained were 93.4% for the coarse classes and 86.2% for the fine classes.

In this case, syntactic and semantic features were used, thus giving better results compared to other question classifiers at that time. However, the hypernyms were not filtered to suit a given word, and instead, the work relied on the SVM to pick the correct hypernyms. The application of WSD during the hypernym feature extraction could further improve the results.

2.2.5 Nguyen, *et al.*'s Question Classifier

Nguyen, *et al*. [19] used two classifiers: the Maximum Entropy Model (MEM) [20] and Boosting Model (BM) [21]. For feature space, the subtree feature was used. The best results reported were 91.2% for the coarse classes and 83.6% for the fine classes.

Only syntactic features (the subtree feature) were used in this classifier. The results are slightly better than that of Zhang & Lee [17] which can be attributed to the use of subtree mining.

2.2.6 Huang, *et al.*'s Question Classification

Huang, *et al*. [2] used two different machine learning approaches: MEM and SVM. For feature space, wh-word, head word, head word hypernyms, n-gram and word shape (or bouma) were used. The headwords were extracted using Collins Head Finder [6] and the hypernyms of these were also extracted to increase the granularity of the feature space. For word shape, features such as word case and digits were considered. The best reported results were 83.6% for the coarse classes and 89.2% for the fine classes. As such, the work reported in this article will be based on Huang, *et al*. [2]'s efforts.

2.2.7 Comparison of Question Classification Approaches

Table 1 compares the results of several question classification approaches with syntactic and semantic features on the coarse and fine classes using the University of Illinois at Urbana-Champaign (UIUC) dataset.

Table 1 Question classification accuracy of related works which also used the UIUC dataset [2].

Algorithm	**Coarse (6 class)**	**Fine (50 class)**
Li and Roth, SNoW	-	78.8
Hacioglu, *et al*., SVM+ECOC	-	80.2 – 82.0
Zhang and Lee, Linear SVM	87.4	79.2
Zhang and Lee, Tree SVM	90	-
Nguyen, *et al*., MEM+BM	91.2	83.6
Krishnan, *et al*., SVM+CRF	93.4	86.2
Huang, *et al*., Linear SVM	93.4	89.2
Huang, *et al*., MEM	93.6	89

2.3 Classifier Features

From Table 1, the best result was achieved by Huang, *et al*. [2]'s work using SVM and MEM. This is due to the use of semantic features such as head word and head word hypernym features. Therefore, the use of semantic features will be further discussed in the next subsections.

2.3.1 Head Word

Li & Roth [10]'s head word extraction method takes the first noun and verb chunk as the head words of a question. In their method, the question is first POS-tagged and then chunked into phrases. From the resulting chunks, the first noun and verb chunk are extracted as the head words.

Krishnan, *et al*. [18] proposed a head word called the informer span. They proposed three approaches to identify the informer span. The first approach is to manually label each question with its informer span resulting in what is called the perfect informer span. The perfect informer span gave the best results of all the three approaches. The second approach utilizes heuristics. This heuristics approach gave the worst results. The third approach is to use Conditional Random Field (CRF) [22] to label informer spans.

Huang, *et al*. [2] extracted head words based on Collins Head Finder [6]. It employs a rule-based approach and it works by checking the context-free rule ($X \rightarrow Y_1 \ldots Y_n$) and determining which of the ($Y_1 \ldots Y_n$) is the head of the rule. The original rules set a higher precedence for verb phrases than noun phrases. This precedence could not be used to determine the correct head word since most head words are noun phrases. Therefore, some rules were modified to suit the circumstances.

2.3.2 Hypernym

In order to identify the correct hypernyms of a word, WSD is often employed. In Huang, *et al*. [2]'s work, they used the Lesk [23]'s algorithm to determine the right sense of a word. Li & Roth [10] and Krishnan, *et al*. [18] did not disambiguate the word. They, instead, extracted all the word senses and left it to the classifier to decide on the correct one. Our work however, uses Adapted Lesk [24]'s algorithm for the WSD.

The Adapted Lesk [24]'s algorithm employs WordNet to disambiguate a given word. It involves defining a window of context surrounding the target word to be disambiguated. The size of the window is n WordNet word tokens to the left and another n tokens to the right for a total of $2n + 1$ words (including the target word). For the algorithm itself, the glosses between each word pair in the window of context are compared. More specifically, the glosses associated with the synset, hypernym, hyponym, holonym, meronym, troponym, and attribute of each word in the pair are compared. The comparisons lead to the identification of overlaps, which contribute to the score of different combination of sense-tags.

The accuracy of the adapted Lesk algorithm is 32% when tested using SENSEVAL-2 data which is higher than the original Lesk [23]'s algorithm accuracy of 23%. Although the state-of-the-art WSD algorithm [25] was able to achieve accuracy as high as 78.1% [26], it was not included in our scope of work in view of its complexity.

2.4 Classifer Engine

This work uses the ELM as its classifier engine. The ELM employs feed-forward neural network architecture and works by randomly choosing the input weight. It then, analytically determines the output weight [4].

Given a training set $\aleph = \{(\mathbf{x}_i, \mathbf{t}_i) \mid \mathbf{x}_i \in \mathbf{R}^n, \mathbf{t}_i \in \mathbf{R}^m, i = 1, \ldots, N\}$, activation function $(g)(x)$, and hidden node number $\tilde{N}$, the ELM algorithm [4] is as shown in Algorithm 1.

Algorithm 1 The ELM Algorithm [4]

```
1: Randomly assign input weight $\mathbf{w}_i$ and bias $\mathbf{b}_i$, $i = 1, \ldots, \tilde{N}$.
2: Calculate the hidden layer output matrix $\mathbf{H}$.
3: Calculate the output weight $\boldsymbol{\beta}$.
```

In benchmarking efforts by Huang, *et al.* [4], the ELM performs much faster than backpropagation and SVM in approximating the sinC function. In a real-world case of diagnosing diabetes using the Pima Indians Diabetes Database, ELM was more accurate compared to SAOCIF, SVM, and Cascade-Correlation. With the ELM's advantage in terms of speed and accuracy, we have employed ELM in our question classification approach.

3 Methodology

Our approach for question classification using ELM is divided into three distinct phases:

- Phase 1: Preprocessing of Training Data
- Phase 2: Question Classification
- Phase 3: Evaluation

3.1 Phase 1: Preprocessing of Training Data

For the purpose of benchmarking, this research uses the UIUC dataset compiled by Li & Roth [10]. This dataset comprises 5,500 training questions and 500 testing questions. The training question set is divided further into 5 different sets of 1,000, 2,000, 3,000, 4,000 and 5,500 questions respectively. Each smaller set is a subset of the larger set. Data distribution for each class (coarse and fine classes) in the 5,500 training question dataset is shown in Table 2.

Table 2 Distribution of the 5,500 (5,452 to be precise) questions that were used in the training phase of question classification.

Class	#	Class	#	Class	#	Class	#
ABBREVIATION	**86**	disease/medicine	103	Term	93	**NUMERIC**	**896**
Abbreviation	16	event	56	vehicle	27	code	9
Expression	70	food	103	word	26	count	363
DESCRIPTION	**1162**	instrument	10	**HUMAN**	**1223**	date	218
definition	421	lang	16	group	189	distance	34
description	274	letter	9	individual	962	money	71
manner	276	other	217	title	25	order	6
reason	191	plant	13	description	47	other	52
ENTITY	**1250**	product	42	**LOCATION**	**835**	period	75
animal	112	religion	4	city	129	percent	27
body	16	sport	62	country	155	speed	9
color	40	substance	41	mountain	21	temp	8
creative	207	symbol	11	other	464	vol.size	13
currency	4	technique	38	state	66	weight	11

3.1.1 Dataset Format Reading

This is a straightforward process that reads through each dataset file and extracts components such as classes and question sentences. All of the training and testing dataset questions have the same format: <CoarseClass>:<FineClass><Question>.

3.1.2 Question Sentence Parsing

The process of question sentence parsing is divided into four steps: tokenizing, tagging, lemmatizing, and chunking. Tokenization step is done by using regular expressions provided by the natural language toolkit (NLTK) library which conforms to the Penn Treebank convention. In this research, only word tokens are needed for the feature extraction process. Other elements, such as punctuations, are discarded.

The next step is tagging step which is done by using POS tagger provided by the NLTK library. The given word is then lemmatized by matching a given word together with its POS tag to its morphological form in WordNet. The last step is chunking which is done through the use of the Berkeley Parser [27]. The parser works by first assigning the correct POS tag to each word in the sentence. The parser then continues chunking the group of words into the Treebank form.

3.2 Phase 2: Question Classification

In this phase, the questions that have been preprocessed are classified into several categories. In order to do this, useful semantic features in the questions are first extracted. These extracted features will then serve as input for the classification process.

3.2.1 Feature Extraction

Generally, in question classification, the extracted features are grammatical properties of a language such as tense, lemma form, POS, and hypernym. Since there are many possible features that can be extracted, only three types of features are used in this research: question wh-word, head word, and hypernym. Question classification efforts by Li & Roth [28] and Huang, *et al*. [2] also used these three features and produced good results.

3.2.1.1 Wh-word

The question wh-word comprises of keywords such as: “what”, “which”, “when”, “where”, “who”, “why”, “how” and “rest”. The “rest” wh-word is for questions that do not belong to any of the other wh-word cases.

3.2.1.2 Head Word

Since Huang, *et al.* [2]'s approach for question classification have shown good results in terms of accuracy using their head word algorithm, this research modified Huang, *et al.* [2]'s head word extraction algorithm to further improve the accuracy. Our modified algorithm is shown in Algorithm 2.

In our modification of Huang, *et al.* [2]'s algorithm, lines 7 to 21, and 46 to 49, introduces new regular expressions into Huang, *et al.* [2]'s list of existing regular expressions. The full list of new regular expressions can be seen in Table 3. Example of questions that are true positive due to these modifications are: "What do bats eat?", "Who was the first governor of Alaska?", "What is the name of vitamin B1?", and so on.

Lines 22 to 45 improves the accuracy of head word extraction through the modification of several parts of the existing algorithm such as prioritizing the "SBARQ" node over other nodes when searching for head words (lines 23 to 29), and differentiating each extraction according to the type of the node (lines 30 to 45). Example of questions that are true positive due to these modifications are: "What kind of animals were in the Paleozoic era?", "What types of water pollution are there?", and so on.

Table 3 New regular expressions added to Huang, *et al.* [2]'s list of regular expressions.

Name of Pattern	Pattern
Hum:ind pattern	The question begin by *Who is/are*
Enty: termeq pattern 2	The question begin with *What is the term*
Enty: termeq pattern 3	The question begin with *What was/is another name*
Enty: food pattern	The question begin with *What do/did/does* and ends with *eat*

Algorithm 2 Modified Huang, *et al.* [2]'s head word extraction algorithm

```
Require: Question q
Ensure: Question head word
1 : if q.type == when|where|why then
2 :     return null
3 : end if
4 : if q.type == how then
5 :     return the word following word "how"
6 : end if
7 : if q.type == what then
8 :     for all regular expression r except HUM:Desc & HUM:Ind pattern do
9 :         if q matches r then
10:             return r.placeholder-word
11:         end if
12:     end for
13: end if
14: if q.type == who && q matches HUM:Desc pattern then
15:     if q matches HUM:Desc pattern then
```

```
16:         return “HUM:Desc”
17:     end if
18:     if q matches HUM:Ind pattern then
19:         return “HUM:Ind”
20:     end if
21: end if
22: Tree tree = question parse tree
23: if tree has node “SBARQ” then
24:     if exist node’s tag starts with “WH” then
25:         Tree whTree = branch of tree with root node’s label starts with “WH”
26:         if whTree first node starts with “NN” then
27:             head = the node that starts with “NN”
28:         end if
29:     end if
30:     while tree is not leaf do
31:         String head = find branch using Collins Head Finder rule
32:         if head is leaf then
33:             if tree label does not start with “NN” then
34:                 head = the first node starts with “NN” in question parse tree
35:             else
36:                 head = head
37:             end if
38:         end if
39:     end while
40:     if tree’s label starts with “Names”|“Types”|“Genres”|“Kinds”|“Groups” then
41:         if tree has branch that match pattern ((PP (IN in) (NP np)) then
42:             head = the NP node in the branch
43:         end if
44:     end if
45: end if
46: if head is upper case && (r.placeholder-word = DESC:def 1 || r.placeholder-word =
DESC:def 2) then
47:     return ABBR:exp
48: end if
49: return head
```

Rules from Collins Head Finder [6] were also modified to give a higher precedence to noun phrases over verb phrases as the chosen head word. Table 4 shows the modification of Collins Head Finder rules [6].

Table 4 Modified Collins Head Finder rules [6].

Parent Non Terminal	Modified Priority List	Original Priority List
FRAG	{"left", "NN", "NNS", "NNP", "NP"}	{rightExceptPunct}
PP	{"right", "NP","IN", "TO", "VBG", "VBN", "RP", "FW", "JJ"}, {"right", "PP"}	{"right", "IN", "TO", "VBG", "VBN", "RP", "FW", "JJ"}, {"right", "PP"}
S	{"left", "TO", "NP", "S", "FRAG", "SBAR","VP", "ADJP", "JJP", "UCP"}	{"left", "TO", "VP", "S", "FRAG", "SBAR", "ADJP", "JJP", "UCP", "NP"}

Parent Non Terminal	Modified Priority List	Original Priority List
SBAR	{"left", "S","WHNP", "WHPP", "WHADVP", "WHADJP", "IN", "DT", "SQ", "SINV", "SBAR", "FRAG"}	{"left", "WHNP", "WHPP", "WHADVP", "WHADJP", "IN", "DT", "S", "SQ", "SINV", "SBAR", "FRAG"}
SBARQ	{"left", "SBARQ","SQ", "S", "SINV", "FRAG"}	{"left", "SQ", "S", "SINV", "SBARQ", "FRAG"}
SINV	{"left", "NP","VP","VBZ", "VBD", "VBP", "VB", "MD", "S", "SINV", "ADJP", "JJP"}	{"left", "VBZ", "VBD", "VBP", "VB", "MD", "VP", "S", "SINV", "ADJP", "JJP", "NP"}
SQ	{"left","NP","ADJP","VP","VBZ", "VBD", "VBP", "VB", "MD", "AUX", "AUXG", "SQ"}	{"left", "VBZ", "VBD", "VBP", "VB", "MD", "AUX", "AUXG", "VP", "SQ"}
UCP	{right}	{"left"}
VP	{"left", "NN", "NNS", "NNP","NP","UCP", "ADJP","PP", "TO", "VBD", "VBN", "MD", "VP", "VBZ", "VB", "VBG", "VBP", "AUX", "AUXG", "JJP", "JJ"}	{"left", "TO", "VBD", "VBN", "MD", "VBZ", "VB", "VBG", "VBP", "VP", "AUX", "AUXG", "ADJP", "JJP", "NN", "NNS", "JJ", "NP", "NNP"}
WHNP	{"left", "NN", "NNS", "NNP","NP", "PP","WDT", "WP", "WP$", "WHADJP", "WHPP", "WHNP"}	{"left", "WDT", "WP", "WP$", "WHADJP", "WHPP", "WHNP"}
WHPP	{"right", "WHNP","IN", "TO", "FW"}	{"right", "IN", "TO", "FW"}
X	{"right", "S", "VP", "ADJP", "JJP", "NP", "SBARQ","SBAR", "PP", "X"}	{"right", "S", "VP", "ADJP", "JJP", "NP", "SBAR", "PP", "X"}
NP	{"rightdis", "NN", "NNP", "NNPS", "NNS", "NX", "JJR"},{"left", "NP", "NML", "PRP"}, {"rightdis", "$", "ADJP", "JJP", "PRN", "FW"}, {"right", "CD"}, {"rightdis", "JJ", "JJS", "RB", "QP", "DT", "WDT", "RBR", "ADVP"}	{"rightdis", "NN", "NNP", "NNPS", "NNS", "NX", "POS", "JJR"}, {"left", "NP", "NML", "PRP"}, {"rightdis", "$", "ADJP", "JJP", "PRN", "FW"}, {"right", "CD"}, {"rightdis", "JJ", "JJS", "RB", "QP", "DT", "WDT", "RBR", "ADVP"}

3.2.1.3 Hypernyms of Head Words

Prior to extracting the hypernyms of a given word, WSD is carried out (see Table 5 for the extracted sense of each head word). The known sense of the word can then be used to search for hypernyms from WordNet. For our purpose, the Adapted Lesk [24]'s algorithm was used for WSD.

After the correct sense of the head words are found, finding the hypernyms is straightforward. The problem lies in determining how deep the level of hypernym that needs to be retrieved. There is no standard way to determine the hypernym depth, but six levels deep are considered sufficient [2]. Table 5 shows the example of feature extraction result using only four questions.

Table 5 Training features for classification.

No.	Question	Wh-Word	Head Word	Head Word Sense	Hypernyms
1	How did serfdom develop in and then leave Russia?	How	Did	Null	Null
2	What are liver enzymes?	What	DESC:def	Null	Null
3	Who killed Gandhi?	Who	Gandhi	political and spiritual leader during India struggle with Great Britain for home rule; an advocate of passive resistance (1869-1948)	leader, person, organism, living thing, whole, object
4	What does a defibrillator do?	What	DESC:desc	Null	null

3.2.2 Classification

ELM was chosen as the classifier for its speed in training and testing. For ELM classification, the first step is to take the features from the feature extraction process and combine those into a feature space. The class code element represents the actual class of each question.

Table 6 Training feature space.

Features	Q1	Q2	Q3	Q4
Class Code	5	3	29	4
What	-1	1	-1	1
Who	-1	-1	1	-1
How	1	-1	-1	-1
did	1	-1	-1	-1
Gandhi	-1	-1	1	-1
DESC:def	-1	1	-1	-1
DESC:desc	-1	-1	-1	1
leader	-1	-1	1	-1
person	-1	-1	1	-1
organism	-1	-1	1	-1
living thing	-1	-1	1	-1
whole	-1	-1	1	-1
object	-1	-1	1	-1

The feature space would be represented as vectors with each vector representing a question. The elements of each vector are the extracted features that are assigned either the digit 1 (indicating that the particular feature occurs in the question) or -1 (the particular feature does not occur). The feature space from Table 5 is shown in Table 6.

3.2.2.1 Training Phase

For the ELM, the training set is defined as $\aleph = \{(\mathbf{x_i}, \mathbf{t_i}) \,|\, \mathbf{x_i} \in \mathbf{R}^n, \mathbf{t_i} \in \mathbf{R}^m, \mathbf{i} = 1, \ldots, \mathrm{N}\}$ from the feature space of questions in Table 6, where matrix **x** represents the input question and matrix **t** represents the actual class.

$$\mathbf{x} = \left(\begin{array}{ccccccccccccccc} (Q1) & -1 & -1 & 1 & 1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ (Q2) & 1 & -1 & -1 & -1 & -1 & 1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ (Q3) & -1 & 1 & -1 & -1 & 1 & -1 & -1 & 1 & 1 & 1 & 1 & 1 & 1 \\ (Q4) & 1 & -1 & -1 & -1 & -1 & -1 & 1 & -1 & -1 & -1 & -1 & -1 & -1 \end{array}\right)$$

and

$$\mathbf{t}(\text{class}) = \begin{pmatrix} 5 \\ 3 \\ 29 \\ 4 \end{pmatrix}$$

Then, matrices with random values for input weights (w) and the biases of the hidden neurons (b) are defined. In this example, the number of hidden neurons chosen is 4, corresponding to the number of training questions. Therefore, the input weight matrix will have four rows for the four hidden nodes and 13 columns for the 13 possible features.

$$\mathbf{w} = \begin{pmatrix} -0.56 & 0.74 & -0.59 & \ldots & -0.84 & 0.48 & -0.12 \\ -0.68 & 0.76 & -0.45 & \ldots & -0.49 & -0.34 & -0.71 \\ -0.67 & 0.93 & 0.92 & \ldots & -1.00 & 0.03 & 0.28 \\ 0.97 & -0.48 & 0.60 & \ldots & 0.09 & 0.54 & -0.03 \end{pmatrix} \quad b = \begin{pmatrix} 0.03 \\ 0.09 \\ 0.11 \\ 0.25 \end{pmatrix}$$

After defining the random input weights and biases, the matrix for the hidden output H is calculated with the equation $\mathbf{H} = \sum_{i=1}^{\tilde{N}} g(w_i \cdot x_j + bi)$. In this research, a linear activation function was used.

Following the calculation of H, the output weight could then be calculated with the equation β = H†T, where H† is the Moore-Penrose generalized inverse of matrix H. The matrix H and resulting β are as follows.

$$\mathbf{H} = \begin{pmatrix} 0.84 & -0.33 & -1.18 & 0.29 \\ 1.63 & 2.02 & -0.92 & 1.83 \\ 2.22 & -0.90 & -0.28 & 1.09 \\ -0.67 & -3.41 & 1.35 & -1.55 \end{pmatrix} \qquad \beta = \begin{pmatrix} -0.04 & 4.72 & -0.75 & -0.51 \\ -4.86 & 17.34 & -9.45 & -0.76 \\ 2.01 & -11.00 & 5.62 & 0.25 \\ -3.69 & 12.98 & -6.70 & -0.17 \end{pmatrix}$$

The calculation of the output weight marks the end of the training phase. The output weight would be used in the classification of the test dataset with the same number of hidden neuron and feature space.

3.2.2.2 Testing phase

As an example to test the ELM's classification, the following question is selected, "What is desktop publishing?". The question features: wh-word, head word, and hypernym, are "what", "DESC:desc", and "null" respectively.

The **x** and **t** matrices for the test question are as follows.

$$x = \begin{pmatrix} 1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \end{pmatrix}$$

and

$$t = \begin{pmatrix} 4 \end{pmatrix}$$

With the same calculation as the training process, the **H** matrix for the test question, **HTest**, is as follows.

$$\mathrm{HTest} = \begin{pmatrix} 0.29 \\ 1.82 \\ 1.08 \\ -1.54 \end{pmatrix}$$

The class of the test question can be predicted using the equation y = **HTestβ**. The **y** for the example test question is as follows.

$$Y = \begin{pmatrix} -1 \\ 1 \\ -1 \\ -1 \end{pmatrix}$$

The result y is the prediction made by the classifier. The predicted class (indicated by the value 1) is in the second row of the matrix. In order to determine which class the second row represents, all available classes from the training questions (see matrix t from the training phase) have to be sorted. The resulting sorted classes are 3, 4, 5, and 29, and the second row corresponds to class 4 which is DESC:desc which is the correct class.

3.3 Phase 3: Evaluation

For evaluation purposes, each of the 5 sets of the data was evaluated on 2 types of classes, namely the coarse classes and fine classes. For each type of classes there are three combinations of features, bringing to a total of 30 settings. The three combinations of features are:

1. wh-word
2. wh-word, and head word
3. wh-word, head word, and (head word) hypernym

The metrics used besides accuracy are precision and recall which were also used by Huang, *et al*. [2]. Precision is the proportion of predicted positive cases that are indeed real positives, while recall is the proportion of real positive cases that were predicted as positive [29]. Speed benchmarking was also done by taking the training and testing time for the coarse and fine classes. These results are then compared to the LibSVM[1] implementation of SVM which is the same SVM engine used in Huang, *et al*. [2]'s work.

4 Results and Analysis

This section presents the results that were obtained and the analysis of the overall and best results.

Table 7 Comparison of the modified Huang, *et al*. [2]'s head word extraction algorithm and Huang, *et al*. [2]'s original algorithm on ELM.

Metric	**Modified Huang, *et al*. [2]'s algorithm**		**Huang, *et al*. [2]'s algorithm**	
	Coarse	Fine	Coarse	Fine
Accuracy	76.8%	70.6%	76.4%	70.7%
# Hidden Nodes	900	1100	900	1100

Table 7 shows the experiment in which the question classification is run using only the head word feature with two versions of implementations: our modified Huang, *et al*. [2]'s head word extraction algorithm and the original algorithm. Both implementations used ELM as the classifier engine. From the results, our modified Huang, *et al*. [2]'s head word extraction algorithm performed slightly better for the coarse class category and slightly worse for the fine class category.

Table 8 shows the results of experiments on the five sets of data. The number of hidden nodes are determined by generate-and-test (trial-and-error) method in which the experiment are run using a very small number of hidden nodes (started at 10) then progressively increasing the number until the best result is achieved.

[1] http://www.csie.ntu.edu.tw/ cjlin/libsvm/

Table 8 Result of experiments on five different dataset

Metric	**Wh-Word**		**Wh-Word + Head Word**		**Wh-Word + Head Word + Hypernym**	
	Coarse	**Fine**	**Coarse**	**Fine**	**Coarse**	**Fine**
1000 questions dataset						
Mean Accuracy (%)	46.29	46.62	85.23	72.34	82.34	72.66
Standard Deviation	0.30	0.20	0.47	0.61	1.43	0.95
Max Accuracy (%)	46.60	46.80	86.40	73.40	86.20	74.40
# Hidden Nodes	10	10	350	350	300	300
#Feature	10	10	457	457	1,003	1,003
2000 questions dataset						
Mean Accuracy (%)	46.00	46.80	88.96	77.70	86.14	77.54
Standard Deviation	0.00	0.00	0.48	0.24	1.19	0.66
Max Accuracy (%)	46.00	46.80	90.00	78.20	88.00	78.80
# Hidden Nodes	10	10	600	600	600	600
#Feature	10	10	782	782	1,547	1,547
3000 questions dataset						
Mean Accuracy (%)	45.60	46.80	89.54	78.61	86.94	78.65
Standard Deviation	0.00	0.00	0.41	0.26	0.99	0.78
Max Accuracy (%)	45.60	46.80	90.60	79.20	89.20	80.60
# Hidden Nodes	10	10	850	850	850	850
#Feature	10	10	1,291	1,291	1,924	1,924
4000 questions dataset						
Mean Accuracy (%)	45.60	46.80	90.77	80.55	88.40	80.17
Standard Deviation	0.00	0.00	0.40	0.24	0.97	0.75
Max Accuracy (%)	45.60	46.80	91.60	81.00	90.40	81.80
# Hidden Nodes	10	10	1150	1150	900	900
#Feature	10	10	1,291	1,291	2,351	2,351
5500 questions dataset						
Mean Accuracy (%)	45.60	46.80	91.65	82.20	89.90	82.98
Standard Deviation	0.00	0.00	0.57	0.33	0.78	0.54
Max Accuracy (%)	45.60	46.80	**92.80**	83.00	91.80	**84.60**
# Hidden Nodes	10	10	1150	1150	1150	1150
#Feature	10	10	1,615	1,615	2,806	2,806

Table 9 shows the precision and recall for each category in the coarse class for the best result (92.80%).

Table 9 Precision and recall for the coarse class

Class	#	Precision	Recall
ABBR	9	100.00%	100.00%
DESC	138	95.71%	97.10%
ENTY	94	76.92%	95.74%
HUM	65	98.36%	92.31%
LOC	81	98.59%	86.42%
NUM	113	99.02%	89.38%

Table 10 shows the precision and recall of each category in the fine class for the best result (84.60%).

Table 10 Precision and recall for the fine class

Class	#	Precision	Recall	Class	#	Precision	Recall
ABBR:abb	1	100.00%	100.00%	ENTY:veh	4	66.67%	50.00%
ABBR:exp	8	100.00%	100.00%	ENTY:word	0	100.00%	100.00%
DESC:def	123	93.75%	97.56%	ENTY:termeq	7	100.00%	71.43%
DESC:desc	7	40.00%	85.71%	HUM:ind	55	89.83%	96.36%
DESC:manner	2	40.00%	100.00%	HUM:title	1	0.00%	0.00%
DESC:reason	6	100.00%	83.33%	HUM:desc	3	100.00%	100.00%
ENTY:animal	16	78.57%	68.75%	HUM:gr	6	37.50%	50.00%
ENTY:body	2	100.00%	100.00%	LOC:country	3	75.00%	100.00%
ENTY:color	10	100.00%	100.00%	LOC:mount	3	100.00%	33.33%
ENTY:cremat	0	100.00%	100.00%	LOC:other	50	83.33%	90.00%
ENTY:currency	6	100.00%	100.00%	LOC:state	7	100.00%	85.71%
ENTY:dismed	2	0.00%	0.00%	LOC:city	18	93.33%	77.78%
ENTY:event	2	50.00%	50.00%	NUM:code	0	100.00%	100.00%
ENTY:food	4	100.00%	50.00%	NUM:count	9	100.00%	77.78%
ENTY:instru	1	100.00%	100.00%	NUM:date	47	100.00%	97.87%
ENTY:lang	2	100.00%	100.00%	NUM:dist	16	100.00%	75.00%
ENTY:letter	0	100.00%	100.00%	NUM:money	3	14.29%	33.33%
ENTY:other	12	25.00%	33.33%	NUM:ord	0	100.00%	100.00%
ENTY:plant	5	100.00%	80.00%	NUM:other	12	75.00%	50.00%
ENTY:product	4	0.00%	0.00%	NUM:period	8	72.73%	100.00%
ENTY:religion	0	100.00%	100.00%	NUM:perc	3	66.67%	66.67%
ENTY:sport	1	100.00%	100.00%	NUM:speed	6	83.33%	83.33%
ENTY:substance	15	88.89%	53.33%	NUM:temp	5	100.00%	80.00%
ENTY:symbol	0	100.00%	100.00%	NUM:volsize	0	100.00%	100.00%
ENTY:techmeth	1	100.00%	100.00%	NUM:weight	4	100.00%	50.00%

4.1.1 Analysis of the Overall Results from the Dataset

From the five sets of the data, Table 8 indicates that the highest accuracies for the coarse classes were for the wh-word + head word features. For the fine classes, the wh-word + head word + hypernym features produced the highest accuracies.

It is also noted that the addition of the hypernym feature seemed to have increased the accuracy for the fine classes, but its inclusion decreased the accuracy for the coarse class. The reason for this is that the extra detail offered by the hypernym feature for the coarse class becomes noise to the classifier. The increase of the size of the sets of data also contributed to the increase in accuracy.

Another observation from Table 8 is that although the size of the feature space for the wh-word + head word + hypernym features is 1.5 times the size of the wh-word + head word features, the number of hidden nodes is actually almost the same. However, for the 1,000 question and 4,000 question sets, the number of hidden nodes is decreasing even though classifying using wh-word + head word + hypernym involved more features than classifying using wh-word + head word. This shows that the hypernym feature does not add much to the complexity of the feature space.

4.2 Analysis of the Best Results

The best results obtained in this experiment in terms of accuracy are 92.80% for the coarse classes and 84.60% for the fine classes. The best result for the coarse classes is better than the state-of-the-art (92.60%) using the same settings. This is due to the improvements on Huang, *et al*. [2]'s head word extraction algorithm and Collins Head Finder [6] rules.

Additions to the list of existing regular expressions also helped in recognizing patterns that were previously not detected. Other improvements such as the modification of existing Collins Head Finder [6] rules and several parts of the head word extraction algorithm gave better result especially in the use of wh-word + head word features. However, the results for the fine classes is slightly lower than the state-of-the-art (85.60%). This is probably due to the additional regular expressions. On the one hand they helped to increase the coarse classes' accuracy, but on the other hand they lowered the granularity of the feature space when used in the classification of fine classes.

From these results, we could make the following deductions:

1. Some regular expressions result in low precision although recall is high. For example, the regular expression which detects the DESC:desc pattern results in a low precision value (40.00%) (i.e. questions of other classes were mistakenly classified as DESC:desc), even though the results have high recall (85.71%) (i.e. many questions are correctly classified). This means that some regular expressions are too specific to the point that they wrongly classify other classes. However, not all regular expressions result with contrasting low precision and high recall. For example, the DESC:def regular expression produces results with a high precision (93.75%) and high recall (97.56%).

2. Poor WSD contributes to low recall. For example, in the case of a question belonging to the ENTY:animal class "What is a baby lion called?", the sense of the head word "lion" was wrongly identified as "the fifth sign of the zodiac". The correct sense is supposed to be "large gregarious predatory feline of Africa and India having a tawny coat with a shaggy mane in the male". Since the retrieved word sense is wrong, subsequently the retrieved hypernyms (region, location, object, physical entity, and entity) were also wrong. This resulted in a wrong classification.
3. Mistakes in the labeling of the training dataset used by Li & Roth [10]. For example, in the case of a question belonging to the ENTY:substance "What is the birthstone for June?", the head word is "birthstone". In the training dataset, there are two questions with the same "birthstone" head word. However, the two questions were classified as ENTY:other and ENTY:def, even though both questions ("What is June's birthstone?" and "What is November's birthstone?") have a similar pattern to the "What is the birthstone for June?" test question.

4.3 Comparison with Huang, *et al*. [2]'s result

The results of question classification using ELM in terms of accuracy and speed are compared to the results obtained by Huang, *et al*. [2]. Huang, *et al*. [2]'s work was chosen because of the machine learning-based statistical question classification approach that was used, as well as their choice of Li & Roth [10]'s dataset. Table 11 and 12 shows the accuracy and speed comparisons between Huang, *et al*. [2]'s results and ELM's results.

Table 11 Comparison of accuracy between Huang, *et al*. [2] and ELM.

Features	**SVM Huang, *et al*. [2]**		**MEM Huang, *et al*. [2]**		**ELM**	
	Coarse	**Fine**	**Coarse**	**Fine**	**Coarse**	**Fine**
wh-Word + Head Word	92.00	81.40	92.20	82.00	**92.80**	83.00
wh-Word + Head Word + Hypernym	**92.60**	85.40	91.80	**85.60**	91.80	**84.60**

From Table 12, ELM achieved a 0.2% increase in accuracy (92.80%) compared to Huang, *et al*. [2] (92.60% for SVM) for the coarse classes.

Table 12 Comparison of speed between SVM and ELM (in seconds).

Class	**Time**	**SVM**	**ELM**	**SVM:ELM**	**% of decrease**
Fine	Training	403.24	37.36	10.79	90.70
	Testing	19.34	1.25	15.47	93.53
Coarse	Training	195.89	33.55	5.84	82.88
	Testing	11.38	0.76	14.97	93.32

For the fine classes, ELM's accuracy (84.60%) was 1.0% lower than the best result of Huang, *et al*. [2] (85.60% for MEM). The cause of the lower accuracy could be attributed to the WSD implementation and the usage of regular expressions. Although the accuracy in the fine classes is lower, it is compensated by an average increase of 92.11% in speed (training and testing) for fine classes.

In the speed comparison, there is a significant improvement compared to Huang, *et al*. [2]'s result. The average training time improvement for all classes is 86.28%, while the average testing time improvement is 93.43%. The average time improvement overall is 90.11%.

5 Conclusion and Future Work

In this research, we employed ELM on semantic features for the purpose of question classification. The use of ELM improves the performance of question classification compared to SVM in terms of speed without sacrificing accuracy. When compared to SVM, ELM performs 5.84 and 10.79 times faster for coarse and fine classes respectively in terms of training time. For testing time, it was 14.89 and 15.47 times faster for coarse and fine classes respectively. This achievement comes without significant difference in accuracy (i.e. 0.2% increase for coarse classes and 1.0% decrease in fine classes).

The addition of four more regular expressions to the existing eight used by Huang, *et al*. [2] reduces the dimensionality of feature space since it introduces fewer words to the feature space by replacing it with a known class-holder string. Smaller dimensionality leads to faster training and testing speeds. However, adding too many regular expressions may reduce the granularity of the feature space and subsequently lower the classifier generality performance (i.e. some test dataset may score very high, while some may score very low).

We have also made some improvements over Huang, *et al*. [2]'s algorithm for head word extraction and Collins Head Finder rules. It can be concluded that the improvements of Huang, *et al*. [2]'s algorithm for head word extraction, Collins Head Finder rules, and regular expressions gave better results for the wh-word + head word feature set. For the wh-word + head word + hypernym feature set, the improvements did increase the accuracy, but the results were comparable without any significant deterioration.

We hope to further improve the accuracy of classification by implementing the n-gram feature. However, including n-gram in the feature space could raise the dimensionality drastically. Hence, an increase in processing capability in terms of CPU processing speed, memory, and perhaps even parallelization is likely to be necessary. More sophisticated data structures for the ELM implementation (e.g. sparse matrix) could also be explored.

References

[1] Riloff, E., Mann, G. & Phillips, W., *Reverse-Engineering Question/Answer Collections from Ordinary Text*, In T. Strzalkowski and S. Harabagiu (Eds.), Advances in Open Domain Question Answering, Volume 32 of Text, Speech and Language Technology, pp. 505–531. Springer Netherlands, 2006.

[2] Huang, Z., Thint, M. & Qin, Z., *Question Classification Using Head Words and Their Hypernyms*, In Proceedings of the Conference on Empirical Methods in Natural Language Processing, Honolulu, Hawaii, Association for Computational Linguistics, pp. 927–936, 2008.

[3] Cortes, C. & Vapnik, V., *Support Vector Network*, Machine Learning, **20**(3), pp. 273–297, 1995.

[4] Huang, G.-B., Zhu, Q.-Y. & Siew, C.-K., *Extreme Learning Machine: Theory and Applications*, Neurocomputing, **70**(1-3), pp. 489–501, 2006.

[5] Rumelhart, D.E., Hinton, G.E. & Williams, R.J., *Learning Representations by Back-Propagating Errors*, In Neurocomputing: Foundations of Research, pp. 696–699, MIT Press, 1988.

[6] Collins, M., *Head-Driven Statistical Models for Natural Language Parsing*, Ph.D. dissertation, University of Pennsylvania, 1999.

[7] Pasca, M.A. & Harabagiu, S.M., *High Performance Question Answering*, Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval, ACM, New Orleans, Louisiana, United States, pp. 366–374, 2001.

[8] Silva, J.a., Coheur, L., Mendes, A. & Wichert, A., *From Symbolic to Sub-Symbolic Information in Question Classification*, Artificial Intelligence Review, **35**, pp. 137-154, 2011.

[9] Fellbaum, C., *WordNet: An Electronic Lexical Database*, Cambridge, Massachusetts: MIT Press, 1998.

[10] Li, X. & Roth, D., *Learning Question Classifiers*, In Proceedings of the 19th international conference on Computational Linguistics, **1**, Taipei, Taiwan, pp. 1–7. Association for Computational Linguistics, 2002.

[11] Even-Zohar, Y. & Roth, D., *A Sequential Model for Multi Class Classification*, In Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing, Pittsburgh, Pennsylvania, pp. 10-19, 2001.

[12] Carlson, A. J., Cumby, C.M., Rosen, J.L. & Roth, D., *Snow User Guide*, Technical Report, Computer Science Department, University of Illinois at Urbana-Champaign, 1999.

[13] Cumby, C. & Roth, D., *Relational Representations That Facilitate Learning*, In Proceedings of the Seventh International Conference on the

Principles of Knowledge Representation and Reasoning, Breckenridge, Colorado, pp. 425-434, 2000.

[14] Hacioglu, K. & Ward, W., *Question Classification with Support Vector Machines and Error Correcting Codes*, In Human Language Technologies 2003: The Conference of the North American Chapter of the Association for Computational Linguistics, Edmonton, Canada, pp. 28-30, 2003.

[15] Dietterich, T.G. & Bakiri, G., *Error-Correcting Output Codes: A General Method for Improving Multiclass Inductive Learning Programs,* In Ninth National Conference on Artificial Intelligence, Anaheim, pp. 527-577, 2002.

[16] Bikel, D.M., Miller, S., Schwartz, R. & Weischedel, R., *Nymble: A High-Performance Learning Name-Finder,* In Proceedings of the Fifth Conference on Applied Natural Language Processing, Washington, DC, Association for Computational Linguistics, pp. 194-201, 1997.

[17] Zhang, D. & Lee, W.S., *Question Classification Using Support Vector Machines*, In Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Toronto, Canada, pp. 26-32, ACM, 2003.

[18] Krishnan, V., Das, S. & Chakrabarti, S., *Enhanced Answer Type Inference from Questions Using Sequential Models*, In Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing, Vancouver, British Columbia, Canada, Association for Computational Linguistics, pp. 315-322, 2005.

[19] Nguyen, M.L., Nguyen, T.T. & Shimazu, A., *Subtree Mining for Question Classification Problem*, In Twentieth Joint International Conference on Artificial Intelligence, Hyderabad, India, pp. 1695-1700, 2007.

[20] Berger, A.L., Pietra, V.J.D. & Pietra, S.A.D., *A Maximum Entropy Approach to Natural Language Processing*, Comput. Linguist., **22** (1), pp. 39-71, 1996.

[21] Schapire, R.E., *A Brief Introduction to Boosting*, International Joint Conference on Artificial Intelligence, pp. 1401-1406, 1999.

[22] Lafferty, J., McCallum, A. & Pereira, F., *Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data*, In Proceedings of the Eighteenth International Conference on Machine Learning, Williamstown, Massachusetts, pp. 282–289, 2001.

[23] Lesk, M., *Automatic Sense Disambiguation Using Machine Readable Dictionaries: How To Tell A Pine Cone from an Ice Cream Cone,* In Proceedings of the 5th Annual International Conference on Systems documentation, Toronto, Ontario, Canada, ACM, pp. 24-26, 1986.

[24] Banerjee, S. & Pedersen, T., *An Adapted Lesk Algorithm for Word Sense Disambiguation Using Wordnet,* In A. Gelbukh (Ed.), Computational

Linguistics and Intelligent Text Processing, Lecture Notes in Computer Science, Springer Berlin/Heidelberg, **2276**, pp. 117-171, 2002.

[25] Yarowsky, D. & Florian, R., *Evaluating Sense Disambiguation across Diverse Parameter Spaces*, Nat. Lang. Eng., **8**(4), pp. 293-310, 2002.

[26] Navigli, R., *Word Sense Disambiguation: A Survey*, ACM Comput. Surv., **41**(2), pp. 1–69, 2009.

[27] Petrov, S. & Klein, D., *Improved Inference for Unlexicalized Parsing,* In Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics, Rochester, New York, pp. 404-411, 2007.

[28] Li, X. & Roth, D., *Learning Question Classifiers: The Role of Semantic Information*, Natural Language Engineering, **12**(3), pp. 229–249, 2006.

[29] Powers, D.M.W., *Evaluation: From Precision, Recall and F-Factor to ROC, Informedness, Markedness & Correlation*, In Journal of Machine Learning Technologies, **2**(1), 2011, pp. 37-63, 2011.