Question Answering
What does this question ask for?
Experiments on Expected Answer Type recognition
Laura Haide Perez

as part of Test Processing course / FUB
January 30, 2009

Abstract
The first stage in a question answering system is question analysis. In order to respond correctly to a natural language question given a large collection of texts, the system needs to understand the question to a level that allows determining some of the constraints the question imposes on a possible answer. Among these constraints is the semantic classification of the sought answer, that is the expected answer type. In this work, we implement two different versions of an expected answer type recogniser; each of them is based on a different approach, namely, a rule based approach and a data driven approach. We describe their design, show their evaluation results and finally compare them.

Contents
1 Project Description 2
1.1 Motivation 2
1.2 Objectives 2
1.3 A brief introduction to question answering 2
1.3.1 Question analysis 3
1.4 Methodology 3

2 Knowledge based approach 4

3 Data driven approach 12
3.1 Results and evaluation 14
3.1.1 Evaluating the features 14
3.1.2 Testing classifiers performance 16

4 Discussion 16


1 Project Description

1.1 Motivation

The first important step in the Question Answering (QA) task is to determine which is the Expected Answer Type (EAT), also referred as the semantic category of the expected answer (e.g., an instance of the concept PERSON). The EAT represents the actual information need expressed by the question itself. Once the EAT is identified, QA systems usually proceed to the following steps of the task, trying to identify, in the target document collection, all the possible entities (candidate answers) with a semantic category compatible with the EAT. At the end of the process, an answer extraction component is usually in charge of filtering out the wrong candidates, and returning the most relevant answer. For example, given the question How long does it take for your blood to make one complete trip through the body?, it is very useful to identify the answer type as a measure, which allows the QA system to narrow the answer search considerably. Thus, the accuracy of the EAT determination step has significant effect on the performance of the QA system.

1.2 Objectives

In this laboratory work we build a QA component that assigns an EAT to an input question posed in natural language. For the construction of this component two different perspectives will be used, a knowledge based approach and a data driven approach.

- Implement a knowledge based version of the QA component for recognising the EAT of a question. This version will be based on simple hand-crafted rules that check for different features of the input questions to assign its appropriate EAT category.
- Implement a data driven version of the QA component for recognising the EAT of a question. An EAT classifier will be trained over a corpus of annotated questions.
- Evaluate the results obtained for each implementation.
- Compare both approaches.
- Practise with basic NLP tools, resources, and techniques (e.g., PoS tagger, WordNet, regular expressions); and with a widely used toolkit for machine learning and data mining (Weka).

1.3 A brief introduction to question answering

Because so much text information is available generally on the web or in specialised collections, the single most important use of language processing these days is to help us query and extract meaning from these large repositories. There are many situations where the user wants a particular piece of information rather than an entire document or document set. The term question answering is used to name the task of returning a particular piece of information to the user in response to questions posed in natural language.

Current general architectures for QA systems typically include a question analysis module that determines the type of question and the type of answer. After the question is analysed, the system typically uses several modules that apply increasingly complex NLP techniques on a gradually reduced amount of text. Thus, a document
retrieval module uses search engines to identify the documents or paragraphs in the
document set that are likely to contain the answer. Subsequently a filter preselects
small text fragments that contain strings of the same type as the expected answer.
Finally, an answer extraction module looks for further clues in the text to determine
if the answer candidate can indeed answer the question.

1.3.1 Question analysis

The first stage within a QA system is question analysis. The input to this stage is a
natural language question. Question analysis involves question classification, extraction
of the expected answer type and focus, extraction of the question context or, more
generally, of semantic constraints which can help the answer extraction.

Most systems carry out a detailed analysis of the question which typically involves
two steps [2]: 1. identifying the semantic type of the entity sought by the question,
that is the **answer type**, (a date, a person, a company, and so on); 2. determining
additional constraints on the answer entity by, for example: (a) identifying key words
in the question which will be used to build a keyword **query** suitable as input to an
IR, and in matching candidate answer sentences; or, (b) identifying relations syntactic
or semantic that ought to hold between a candidate answer entity and other entities
or events mentioned in the question.

In general, QA systems have built hierarchies of question types based on the types
of answer sought, and attempt to place the input question into the appropriate cat-
genory in the hierarchy. The results of the question analysis module for identifying
the questions model has a greater effect on the rest of the QA system. Once the
type of entity being sought has been identified, the remaining task of question analysis
is to identify additional constraints that entities matching the type description must
also meet. This process may be as simple as extracting keywords from the rest of
the question to be used in matching against candidate answer sentences. This set of
keywords may then be expanded, using synonyms and/or morphological variants or
using full-blown query expansion techniques by, for example, issuing a query based
on the keywords against an encyclopedia and using top ranked retrieved passages to
expand the keyword set. Or, the constraint identification process may involve parsing
the question with grammars of varying sophistication.

1.4 Methodology

The question type is used to define the **expected answer type**, which means the type of
information that is being sought. For example, one strong indicator of question type
is provided by the question words (WH-terms or interrogative pronouns), such as,
*When*, *Whose*, *Where*, *Why*. Additional mechanisms and resources for identifying
the question type are also necessary, because some of the question words are ambiguous
or do not provide enough clues to indicate the question type (e.g. *What*). Then, its
necessary to look for other information given by the rest of the terms in the question.

In our two versions of the EAT recogniser component, rule based and data driven
approaches, we aim at classifying questions based on syntactic and semantic informa-
tion gleaned from some tokens present on them. For instance, the question *Who wrote
Hamlet?* has EAT type PERSON and we can deduce this from its first Wh-term *Who.*
Or in the question *what character dances with Mia?* the EAT is not revealed by *what*
alone, here the syntax and information of the second term, a noun that is hyponym of
PERSON, help to reveal the EAT of the question. In the ruled based EAT recogniser
this methodology was applied when writing the rules whereas for the data driven EAT
recogniser it was applied to design the features that will train the classifier.
Our models were built based on the question samples in the training data sets. These questions include traditional WH-terms, which begin with what, when, where, which, who, why and how, as well as imperative statements starting with name. Some questions expect an entity belonging to semantic classes of words (e.g. PERSON, LOCATION, ANIMAL) while other questions look for other types of answer like definition, reason, description, manner. Our EAT recognisers use the question classification taxonomy shown in Table 1. In both implementations of the EAT recognizer component we use WordNet to verify if a given noun is a hyponym of some of the concepts naming the semantic classes in this classification taxonomy.

In the following sections, we describe how the analysis of the question is done within the implementation of both the knowledge based and data driven EAT recognisers. We explain the written rules and the designed features.

2 Knowledge based approach

In our project we build a ruled-based EAT recogniser on the existing backbone of the EAT recognition component called frediogene. More precisely, our work consist in developing a set of rules by extending an existing one, with sample rules, in order to cover as many training examples as possible.

The strategy we adopted was to incrementally build the set of rules. In each incremental step, we enrich the set with new rules and evaluated it to see the amount of improvement obtained. After each evaluation, in the case of discovering errors in the classification, we modified the rules to solve them and in some cases we went through and modify the lists of hyponymy terms acquired from WordNet (that is adding or deleting some terms).

As a first step in the construction of the set of rules, we focused on writing both i) simple forms of rules and ii) rules for the most frequent EAT classes in the training data set. Therefore, we wrote rules for those questions that contain WH-terms which carry semantic typing information, such as, the WH-term When that seeks a DATE entity as answer; Where a LOCATION; Who a PERSON. As each type of WH-term looks for different types of answer we used separate rules for each question type. And we wrote rules for questions with PERSON and LOCATION EATs. Following, we describe the rules and for each of them we show a sample question extracted from the training data set covered by the rule.

<table>
<thead>
<tr>
<th>Rule Name</th>
<th>Context</th>
<th>Question</th>
<th>EAT</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>who1</td>
<td>simple</td>
<td>who directed Pulp Fiction?</td>
<td>PERSON</td>
<td>who directed Pulp Fiction?</td>
</tr>
<tr>
<td>date1</td>
<td>simple</td>
<td>When did Iraqi troops invade Kuwait ?</td>
<td>DATE</td>
<td>When did Iraqi troops invade Kuwait ?</td>
</tr>
<tr>
<td>loc1</td>
<td>simple</td>
<td>where am I?</td>
<td>DATE</td>
<td>where am I?</td>
</tr>
<tr>
<td>reason1</td>
<td>simple</td>
<td>Why do recipe books recommend starting with cold water when you boil something?</td>
<td>DATE</td>
<td>Why do recipe books recommend starting with cold water when you boil something?</td>
</tr>
</tbody>
</table>

Various questions in the training set involve English question words, such as Which and What that are not enough to determine the EAT of the question. Thus, hereinafter the added rules consider further semantic information given by some terms or certain syntactic patterns present in the questions that define characteristics of the expected answer type.

In addition, within the first step, we came up with some more detailed rules for recognising question types which look for the most frequent EAT classes appearing in
the training data set, namely, PERSON and LOCATION.

For the questions containing the What or Which question words, the rules we have written look at the noun phrase that follows them, trying to identify the noun which may be considered as the focus. This is the term that gives the semantic class for the question’s EAT. For discovering the semantic class of this terms a list of semantically related words is used. We generate that list for most of the semantic classes used by our EAT recogniser. The terms in these lists are acquired from WordNet hyponym relations. For instance, a list of terms referring to a person is created by extracting from WordNet all the terms that are hyponyms of the word PERSON. Then, the working of this rules is the following, we use a variable associated with a predicate which is in charged of verifying if the given term belongs to the list.

The rules for PERSON are:

This rule was given in the initial data set, it matches with any question starting with a word whose lemma is “what”, followed by a term referring to a person, whose PoS satisfies predicate noun-p associated to variable ?%noun.

(1) What character dances with Mia?

(defmatch-rule what-who1 :simple :context :QT
  (((? ?%what ? ?)) ?)
  (((? ?%person-common-name ?%noun ?)) ?) ~)
("PERSON")

The following rule matches with any question starting with a word whose lemma is “what”, followed by any term that is an adjective, followed by a term that satisfies the predicate associated to the variable %person-common-name, that should be a term referring to a person, and that has a PoS that verifies the predicate associated to variable ?%noun.

(2) what very funny character dances with Mia?

(defmatch-rule what-who-3 :restricted :context :QT
  (((? ?%what-which ? ?)) ?)
  
  as
  (((? %person-common-name ?%noun ?)) ??)
  ("PERSON")
  #'(lambda (dict) (every #'(lambda (x) (adjective-p (third (caar x)))) (lookup "aa" dict)))
)

The rule in (3) matches with any question starting with a word whose lemma is what, followed by a term with lemma be, followed by the determiner the. Then any term that is an adjective, followed by a term that satisfies the predicate associated to the variable %person-common-name, that is a term referring to a person, and that has a PoS that verifies the predicate associated to variable ?%noun.

For this rule we could have used a predicate to skip all the terms after the What word that are not nouns. But we wanted to differentiate from the questions where the determiner a is present as it would be an indication of a question looking for a DEFINITION.

(3) What was the infamous pseudonym of Peter Sutcliffe ?

(defmatch-rule per4 :restricted :context :QT
  ( (((? ?%what-which ? ?)) ?) (((? "be" ? ?)) ?) (((? "the" "DT" ?)) ?)
  as
  (((? %person-common-name ?%noun ?)) ??)
  ("PERSON")
  #'(lambda (dict) (every #'(lambda (x) (adjective-p (third (caar x)))) (lookup "aa" dict)))
)
The next two rules (4 and 5) also match What questions but they look for a LOCATION semantic class. They are the same as the ones for PERSON, the only thing that varies in these rules is the use of a different variable that associates to a different predicate, checking in the list of words that are hyponyms of LOCATION. Namely %location1-common-name and its corresponding predicate location1-common-name-p. It is important to notice here, that the list of location common names was acquired from WordNet as the hyponyms of i) the first sense of the term location, ii) the first sense of the term body of water, and iii) the first sense of the term celestial body.

(4) What province is Edmonton located in?
(defmatch-rule loc3 :simple :context :QT
(((((? %what-which ? ?)) ?) (((? %location1-common-name ?%noun ?)) ?) "")
("LOCATION"))

(5) What are the only two states that incorporate the Confederate battle flag in their flags?
(defmatch-rule loc4 :restricted :context :QT
(((? %what-which ? ?)) ?) (((? "be" ? ?)) ?) (((? "the" ? ?)) ?)
"aa
(((? %location1-common-name ?%noun ?)) ?) 
("LOCATION")
#
(lambda (dict) (every #'(lambda (x) (adjective-p (third (caar x)))) (lookup "aa dict))))
)

The last rule added in this first step is slightly different, but again it aims at detecting questions with LOCATION EAT. It matches with questions starting with the In which or In what sequence of words. Then, we verify that the noun following any of these sequences should be a location noun.

(6) In which city I don’t want to live?
(defmatch-rule loc2 :simple :context :QT
(((? "in" ? ?)) ?) (((? %what-which ? ?)) ?) (((? %location1-common-name ?%noun ?)) ?) 
("LOCATION")
)

#'(lambda (dict) (every #'(lambda (x) (adjective-p (third (caar x)))) (lookup "aa dict))))

Our initial version of the set of rules gave the following results:
Precision: 0.49473685
Recall: 0.46534654
F-measure: 0.47959188

In the second step, we extended the set by adding the same “What” questions’ rules introduced previously but that classify question’s EAT types in the ANIMAL and EVENT semantic classes.

(7) What four-legged creature did a Cornell University study say would make man’s best companion in space?
(defmatch-rule anim1 :restricted :context :QT
(((? %what-which ? ?)) ?)
"aa
(((? %animal1-common-name ?%noun ?)) ?) 
("ANIMAL")
#
(lambda (dict) (every #'(lambda (x) (adjective-p (third (caar x)))) (lookup "aa dict))))
)

As in the training set the noun creature is associated to the ANIMAL expected answer type, we deleted it from the persons common names list obtained from WordNet. If it were not deleted, then for the question in this example we would get the EAT PERSON, as the rules for this eat are placed before than the ones for ANIMAL.
What tragedy befell the city of Dogtown in 1899?

The difference in each rule is the variables and predicates used for them. For the first the pair %animal1-common-name variable and animal1-common-name-p, and for the second one the pair %event1-common-name variable and event1-common-name-p.

The results after testing the EAT recogniser only improve a little giving the following measures:

Precision: 0.51578945
Recall: 0.48514852
F-measure: 0.5

For the next steps, the third and fourth version of our set of rules, we focused on questions that involve a syntactically slightly more complex construction. And we extended some sets of rules with some specific semantic categories, and incorporate new sets of rules for semantic categories not considered so far.

The following rule, matches the questions that start with the pronoun Whose, the possessive case of Who. As the rule for who, its very simple because the semantic information carried by the question word is enough:

Whose video is titled Shape Up with Arnold?

For the questions that seek for a MEASURE we build the following rules. The first rule matches the questions that start with the WH-term How and are followed by a term that is an adjective indicating measure. The variable %qmeasure is associated to the predicate qmeasure-p which verifies if the term belongs to a hand-crafted list. We list these terms without using WordNet, as they are scattered along many WordNet terms, in fact, such list is not so long and even we include few adjectives but could be extended.

How long was Mao’s 1930s Long March?

For the questions that seek for a MEASURE we build the following rules. The first rule matches the questions that start with the WH-term How and are followed by a term that is an adjective indicating measure. The variable %qmeasure is associated to the predicate qmeasure-p which verifies if the term belongs to a hand-crafted list. We list these terms without using WordNet, as they are scattered along many WordNet terms, in fact, such list is not so long and even we include few adjectives but could be extended.

How large is the Arctic refuge to preserve unique wildlife and wilderness value on Alaska’s north coast?

Following, the rules for questions of the type What that seek for a MEASURE answer. The first one, matches with any question starting with a word whose lemma is what, followed by a term with lemma be, followed by the determiner the. Then, any term that is an adjective, followed by a term that satisfies the predicate associated to the variable %measure2-common-name, that is a term referring to a measure common name for the hyponyms of second sense of the term measure in WordNet, and that has a PoS that verifies the predicate associated to variable %noun.
What is the average age of a member of the team that worked on the Manhattan Project?

(defmatch-rule measure2 :restricted :context :QT

((((? %what ? ?) ?) 
((? "be" ? ?) ?) 
((? "the" "DT" ?) ?) 
"aa

(((? %measure2-common-name ?%noun ?)) ?) 
"MEASURE"

#'(lambda (dict) (every #'(lambda (x) (adjective-p (third (caar x)))) (lookup "aa dict))) 
)

The following two rules vary in considering information of terms after the lemma be. The first one, matches with a term with a possessive pronoun PoS, and then any number of adjectives followed by a term referring to a measure, its second sense's hyponyms in WordNet, whose PoS satisfies the predicate noun-p.

(defmatch-rule measure3 :restricted :context :QT

((((? %what ? ?) ?) 
((? "be" ? ?) ?) 
((? "PP$" ?) ?) 
"aa

(((? %measure2-common-name ?%noun ?)) ?) 
"MEASURE"

#'(lambda (dict) (every #'(lambda (x) (adjective-p (third (caar x)))) (lookup "aa dict))) 
)

The second, matches with a term with a proper name PoS, followed by a possessive marker (’s), and then any number of adjectives followed by a term referring to a measure, whose PoS satisfies the predicate noun-p.

(defmatch-rule measure4 :restricted :context :QT

((((? %what ? ?) ?) 
((? "be" ? ?) ?) 
((? "NP" ?) ?) 
((? "POS" ?) ?) 
"aa

(((? %measure2-common-name ?%noun ?)) ?) 
"MEASURE"

#'(lambda (dict) (every #'(lambda (x) (adjective-p (third (caar x)))) (lookup "aa dict))) 
)

Regarding question with MONEY EAT, we write the following rules. The first rule matches questions that start with the word whose lemma is how, followed by a term that is the adjective of quantity with PoS “RB”, in turn, followed by any number of terms of any type, followed by a term referring to a noun hyponymy of cost for its first sense in WordNet and whose PoS satisfies the predicate noun-p.

(12) How much did the minimum wage amount to in 1991?

(defmatch-rule money1 :simple :context :QT

((((? %how ? ?) ?) 
((? "RB" ?) ?) 
(?) 
((? %cost1-common-name ?%noun ?)) ?) 
"MONEY")

8
This second version of the rule is because we got the list of semantically related words for the semantic class MONEY from two different definitions from WordNet. Thus, we generated 1 two different text files and built two predicates for verifying on both of them. Another possible solution could have been merging the list of words of this two files to generate only one file, list of words, for the semantic class MONEY. The reason we took two different general terms for building the lists is because, in the training data set, questions with the term “wage” as well as the term “income” have MONEY EAT. However, it was not possible to get them as hyponyms of “money”. As a result we looked up their definitions 2 in WordNet and found that they have a difference in meaning, thus, are not hyponyms of the same most general concept. Therefore, we selected “cost” and “financial gain” as most general concepts from which to take the hyponyms.

(defmatch-rule money2 :simple :context :QT
  (((? ?%how ? ?)) ?)
  (((? "RB" ?)) ?)
  
  (((? ?%financial-gain1-common-name ?%noun ?)) ? "
  "MONEY"))

What is the per-capita income of Colombia, South America, ?

(defmatch-rule money3 :restricted :context :QT
  (((? ?%what ? ?)) ?)
  (((? "be" ? ?)) ?)
  (((? the" ? ?)) ?)
  aa
  (((? %cost1-common-name ?%noun ?)) ?) "
  "MONEY")
  #'(lambda (dict) (every #'(lambda (x) (adjective-p (third (caar x)))) (lookup "aa dict))))
)

What is your income?

(defmatch-rule money4 :restricted :context :QT
  (((? ?%what ? ?)) ?)
  (((? "be" ? ?)) ?)
  (((? "PP$" ?)) ?)
  aa
  (((? %cost1-common-name ?%noun ?)) ?) "
  "MONEY")
  #'(lambda (dict) (every #'(lambda (x) (adjective-p (third (caar x)))) (lookup "aa dict))))
)

What is John’s income?

(defmatch-rule money5 :restricted :context :QT
  (((? ?%what ? ?)) ?)
  (((? "be" ? ?)) ?)
  (((? "NP" ?)) ?)
  (((? "POS" ?)) ?)
  aa
  (((? %cost1-common-name ?%noun ?)) ?) "
  "MONEY")
  #'(lambda (dict) (every #'(lambda (x) (adjective-p (third (caar x)))) (lookup "aa dict))))
)

1 By using the command: get-hypo -c cost n 1 and get-hypo -c financial gain n 1
2 1. wage, noun. 1. wage, pay, earnings, remuneration, salary, something that remunerates; "wages were paid by check"; "he wasted his pay on drink"; "they saved a quarter of all their earnings"
The results of our third version of the set of rules:

Precision: 0.5555556
Recall: 0.5445545
F-measure: 0.5500001

The fourth incremental step over our set of rules incorporate the rules for What WH-term that have ORGANIZATION and DEFINITION EAT.

For questions with ORGANIZATION EAT, we considered the imperative statements starting with name. The following rules match with a statement that begins with the term whose lemma is name then can be followed by either a sequence of adjectives or a determiner and a sequence of adjectives. Following, there should be a term that is referring to an hyponymy of organization and with noun as PoS. Finally, followed by a sequence of zero or more terms.

(16) Name the various super-teams to which the Angel has belonged.

(defmatch-rule org1 :restricted :context :QT

( (((? ?%name-syn ? ?)) ?) (((? ? "DT" ?)) ?)

"aa

(((? ?%organization1-common-name ?%noun ?)) ?) ~)

("ORGANIZATION")

#'(lambda (dict) (every #'(lambda (x) (adjective-p (third (caar x)))) (lookup "aa dict)))

)

(defmatch-rule org11 :restricted :context :QT

( (((? ?%name-syn ? ?)) ?) (((? ? "DT" ?)) ?)

"aa

(((? ?%organization1-common-name ?%noun ?)) ?) ~)

("ORGANIZATION")

#'(lambda (dict) (every #'(lambda (x) (adjective-p (third (caar x)))) (lookup "aa dict)))

)

As to the rules starting with what but with ORGANIZATION EAT, they have similar structure as the previously introduced ones. They consider the pattern What followed by a noun phrase whose head noun is an hyponym of the term naming the semantic class (in this case ORGANIZATION).

(17) What Japanese manufacturer is known for both its pianos and its motorcycles?
What was the backup singing group for Roy Rogers?

When evaluating our set of rules we got the following classification:

PERSON—ORGANIZATION—What Japanese manufacturer is known for both its pianos and its motorcycles?

As manufacturer has two senses one as organization and the other as person (someone). Then the question was classified as PERSON because we have placed the person rules before the organization ones. To resolve this ambiguity, as in WordNet the first sense is regarding to the organization sense, we eliminated from the person common names the lemma manufacturer.

To the previously simple rule for identifying questions looking for a DEFINITION as answer type, we added the following ones all of them starting with the Wh-term what. The first one is not using the indefinite determiner a to ask for a definition, instead is using the definite because it is asking for an entity that has a known proper name, but still is expecting for a definition. For the following particular question example, we deleted “Immaculate Conception” from the measure common names list. The tagger also tags this separately immaculate as adjective and Conception as noun.

What is the Immaculate Conception?

The same for this rule for questions which directly ask for a named entity.

What is Candlemas Day?
Finally, the following rule for questions that use quotations to enclose the entity for which it is asking a definition. It matches the questions that start with a term whose lemma is *what*, followed by a term whose lemma is *be*, followed by opening quotations, then, zero or more terms, and closing quotations and the interrogation symbol.

(21) *What is "the washed vodka"?*

```
(defmatch-rule def4 :simple :context :QT
  ( ((((? ?what ? ?)) ?) (((? "be" ? ?)) ?)
  (((? ? """" ?)) ?)
  (((? """" ?) ?) )
  ("DEFINITION")
  #'(lambda (dict) (every #'(lambda (x) (noun-p (third (caar x)))) (lookup '"nn dict))))
)
```

Our last version results:

Precision: 0.63  
Recall: 0.62376237  
F-measure: 0.6268657

3 Data driven approach

In this approach, each question should be analysed and represented as a list of features in order to be treated as a training example for the task of learning an EAT classifier. To do this, we designed a set of relevant features in order to describe our training data. We make use of the WEKA machine learning package (Witten and Frank, 1999) to train an EAT classifier using different learning algorithms that could be appropriated for our learning task (e.g. Decision Tree, Boosting, Bagging, Naïve Bayes). For the generation of the ARFF [5] file required by WEKA we wrote a program in Java (Appendix 4).

Among the designed features there are both syntactic feature such as PoS tags or some chunks or sequences of PoS and lemmas; and semantic features based on words’ semantics, such as the semantic class of nouns geared from hyponymy relations in WordNet; or the presence of some words semantically related to the questions’ classes which could include nouns as well as verbs or adjectives.

The set of attributes designed for our EAT recognition learning task is the following:

1. The *firstWord* attribute simply represents the first word in the sentence (e.g. *What*, *Who*, *Name*). For this attribute, we extract the lemma. As we mention in Section 1.4, some of the first words such as *who* (asking for what person or persons?) carry some semantic meaning.

2. The attributes *POSSecondWord* and *LemmaSecondWord* represent the values of the PoS and lemma, respectively, of the second words in the question. For instance, for the question *What country is home to Heineken beer?*, for the word *country* we take NN and country as values for both attributes.

3. Then the attributes *POSThirdWord*, *POSFourthWord* and *POSFifthWord*. They take respectively the PoS of the third, fourth and fifth words in the question. For the example given before the attributes should have values VBZ (*is*), NN (*home*) and TO (*to*).

4. Attribute *ContainsQuotations* has two possible values indicating whether the questions contains or not some quoted substring. For the case of the previous
example the value of the attribute would be negative (value equal to 0), and for
the case of the question What is "dew point"? the value of the attribute would
be positive (value equal to 1).

5. FirstNounComonName is an attribute having one of the following possible values:
person, location, animal, measure, money, event, organization, date, other. The
value for this attribute is extracted from the semantic class of the first noun
in the sentence. More precisely, for the first noun of the sentence we obtain its
semantic class by looking into hyponymy relations in WordNet. We just consider
the semantic classes for the values listed here, but it could be possible to include
other semantic classes. For instance, for the question What country is home to
Heineken beer? the value of this attribute would be location.

6. LemmaFirstNounComonName in the case of this attribute the values will be di-
rectly the lemmas of the first noun. This attribute aims at complementing the
information given by the previously introduced attribute. In the case where the
category was not found, the classifier could learn from the concrete noun words.

7. The binary attribute LocationPP aims at pointing out the presence of preposi-
tions of location in the questions. The possible values here would be 1 in the case
of the presence of a preposition of location or 0 in the contrary case. Here, we
use a hand-crafted list of prepositions of location (e.g at, on, under, near). For
the question What U.S. state lived under six flags? the attribute value would
be 1.

8. DefinitionRelatedWords is a binary attribute whose value could be 1 or 0
based on whether the question contains or not at least one word contained in a
hand-crafted list with words semantically related to questions related to EAT
DEFINITION (e.g. mean, meaning, define, definition). For instance, the value
of the attribute for the sentence What is the meaning of Jesus? would be 1.

9. The nominal attribute HowManyMuch takes one of the following values: many,
much, or none, depending on whether within the sentence it is present the se-
quence How many, How much; or none of them.

10. The attribute ProperName may have 1 or 0 values depending on whether its
present in the question a proper name. For the sample question What is a film
starring Jude Law? this attribute would have value equal to 1, and for the
question How many years do fossils take to form? the attribute contains value
0.

11. The binary attribute ToBeDTSequence takes 1 value when the sequence “to
be” followed by “DT” is present in the question and 0 on the other case. For
instance, in the sentence What is the origin of the world? the value of this
attribute should be 1.

12. The binary attribute HowToBeMDSequence is true (value 1) when the sample
question contains the sequence How followed by to be or by “MD”, and false
(value 0) on the other case.

13. The binary attribute AbbreviationsName is true (value 1) when the sample ques-
tion contains a word with all capitalised letters or with dots. For instance, in
the sample sentence What does S.O.S. stand for? the attribute has value 1.

3 this list of prepositions is stored in a text file named prepositionsOfLocation
4 this list is of words is stored in a text file named DefinitionSemanticRelatedWords
3.1 Results and evaluation

3.1.1 Evaluating the features

In our learning task, instances are described by a fixed set of features and their values, and the target function has discrete output values (class feature with values in Table 1). Therefore, decision tree learning methods are suitable for our classification problem [4]. In order to evaluate the set of features we used the decision tree learning algorithm J48 (C4.5), available in WEKA package, with a cross-validation option set to 10.

First, we tried with the whole set of designed features, described in the previous section, and we got a 68.6928% of correctly classified instances. Then, we tried with different subsets of features with the purpose of evaluating each feature’s performance or contribution. We used the Preprocess Tab in WEKA, for selecting the features subsets.

The most powerful of those features are firstWord, POSSecondWord and LemmaSecondWord. Considering only this attributes after running the classifier it achieve 57.2218% of correctly classified instances. From the observation of the detailed accuracy by class we could see that for those semantic classes related to first words which carry enough semantic information to determine the class, the algorithm obtained some relatively high f-measure values. In addition, from the PoS and the lemma of the second word the algorithm gets information to learn which attribute values distinguish among some given classes. For example, it learns that when the attribute firstWord=how and the LemmaSecondWord=many the class should be CARDINAL, but when the LemmaSecondWord=can (or some other modal verb with “MD” PoS) the class should be MANNER.

Nevertheless, for other classes the f-measure values obtained were very low, so that means that more information, some additional attributes, is needed in order to distinguish the questions EAT for those classes. If we add the features POSThirdWord, POSFourthWord and POSFifthWord to the ones used before after evaluating them we got a 60.8232% of correctly classified instances. Here we observed that the improvement was not that much, and that is because we only add syntactic information. This only helps in learning a bit more about syntactic patterns or sequences in different kinds of questions, but not enough to desambiguate, that is there is a need of other kind of information like the words, lemmas and its semantics.

In the next evaluation we incorporate the features FirstNounComonName and LemmaFirstNounComonName. This two attributes improve the classifier performance, eventhough we are just considering the first noun in the question and not the head noun in the noun phrase. In general, in factual questions it is the focus and it gives the semantic class type of the question’s answer [3]. The percentage of correctly classified instances is 68.6166%.

Then, we incorporate, to the current set of features being evaluated, the attribute DefinitionRelatedWords, this attribute produced a small increment in the accuracy. Until here we can observe the evaluation results obtained at each of the four evaluations steps, in Figure 1.

Following, we added the attributes HowMuchMany and HowToBeMDSequence they did not produce any change in the results, that is because the sequences How many, How much and How MD or How ToBe are being distinguished by the classifier through the attributes firstWord, POSSecondWord and LemmaSecondWord. The same situation holds for the attribute ToBe_DT_Sequence as many noun phrases within questions contain this sequence.

Finally, we observed, after adding for evaluations the following attributes, that they are not good ones. For example, the attribute ProperName, considering the way
it works and the information it gives, is not useful for disambiguate or learn and important feature value for a given EAT class, as questions with different EATs could have proper names. The same occurs with the attribute ContainsQuotations. Regarding the incorporation of the attribute LocationPP, as many questions with different kind of EAT contain this binary features with value 1, and questions with LOCATION EAT have this feature value equal to 0, the algorithm do not obtain useful information.

Even, considering its values in combination with some other attribute values.

After all this previous evaluations of different subsets of features, we decide to consider a smaller set of features for training the classifier, eliminating the features that do not provide enough distinguishing information. Consequently, we got even a better result, 70.6364% correctly classified instances, with the set of 8 features: firstWord, POSSecondWord, LemmaSecondWord, FirstNounComonName, LemmaFirstNounComonName, DefinitionRelatedWords, HowManyMuch, ToBe_DT_Sequence, and AbbreviationsName. After this, we extended the defined possible nominal values for the feature FirstNounComonName. We added the following values: title, plant, food, profession, substance and sport. Accordingly, the first noun is also checked for membership to the lists of words from the hyponyms relations in WordNet corresponding to these added general concepts. As a result, we got a 70.9985% of correctly classified instances.

We note that the decision tree always splits first on the initial component of a question, e.g., firstWord feature, and that one of the second-split feature is the LemmaSecondWord following the initial component.

Another observation comes from the application of some filters with WEKAs preprocess functionalities. To the set of best 9 features we applied the filter Supervised→Attribute→NominalToBinary which made a transformation of the nominal features into binary features. Then, we applied again the J48 classifier and we got a similar result, 69.9505% in the percentage of correctly classified instances.

On the other hand, we did some evaluations with other algorithms. The set of 9 features that better performed with the previous evaluation, did also well with the DesicionTable and Bagging algorithms, with 62.7287% and 69.4931% of correct classified instances, respectively. Instead, the Naive Bayes algorithm did better considering the whole set of features.

At the beginning, we implement the attributes firstWord, POSSecondWord, LemmaSecondWord, POSThirdWord, POSFourthWord, POSFifthWord as nominals. Then, we transformed them into NUMERIC attributes by implementing
dictionaries to store the values for this features. We then use the position of the terms within the dictionary as the features values. This change gave us the flexibility to evaluate different test sets, without having the problem of the new words not declared in the nominals header list.

3.1.2 Testing classifiers performance

To test our classifier, we first generated the ARFF [5] file. We chose as classifier, the one that had better performance, that is the classifier built with the J48 algorithm and the set of 9 best features, described in the previous section. As a result, for the test option we got a 74.0196% of correctly classified instances.

4 Discussion

Both approaches exploit clues such as the first WH-term (e.g. who, where, when, how) and the semantic class of the first noun, among others. For the case of the first noun attribute, we observed that, if instead of considering the first noun we considered the head noun of the noun phrases associated with the main verb, we could improve the accuracy in both strategies. Some noun phrases may contain compound nouns of the form [word word ... head-noun], where word can be a noun or an adjective, such as, [singing group] or [famous singing group]. Another shortcoming in the implementation of both approaches regarding finding the head noun is the case of sentences with possessive marker “’s”. For instance, in the sentence What was the name of Roy Rogers’s dog? with EAT ANIMAL, the term dog should be the noun considered to get the EAT of the question.

Both technologies suffer the ambiguity problem in the question classification task, that is, there is no completely clear boundary between classes. Therefore, the classification of a specic question can be quite ambiguous. For instance, What is the PH scale? could be a numeric value or a denition. It is hard to categorise those questions into one single class. Another ambiguity problem, present in both approaches, is the ambiguity given by some noun terms which occur as hyponymy of different general terms associated to different semantic classes. For example, the noun creature occurs in both PERSON and ANIMAL hyponyms.

However, we found machine learning method for question classification to be advantageous over the rule based method. First, the construction of a rule based classifier for questions is a time consuming task that requires the analysis of a large number of questions. In the rule based approach we wrote different rules for different questions starting with different WH-terms (e.g. who, when, why), while in the machinery learning approach the feature FirstWord accounts for this WH-terms and the algorithm can learn a classification based on its values.

Moreover, mapping questions into fine classes requires the use of lexical items (specic words) and therefore for an explicit representation of the mappings we may get a very large set of rules. On the other hand, in our learning approach one can define only a small number of features that on its values they take into account those specific lexical items. In addition, if we consider reformulations that target the same answer type, different words and syntactic structures make it difficult for a rule based classifier based on a small set of rules to generalise well and map all these to the same answer type, so again a large set of rules is required. Learning methods, on the other hand, with appropriate features may account for paraphrases and would generalise and classify them correctly. Finally, a learnt classifier is more exible to reconstruct than a rule based one because it can be trained on a new taxonomy in a very short time.
References


Overview of major scripts

Question classification taxonomy

<table>
<thead>
<tr>
<th>TAG</th>
<th>DESCRIPTION</th>
<th>EXAMPLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABBR-ABBREVIATION#</td>
<td>abbreviation</td>
<td>What is the abbreviation for micro?</td>
</tr>
<tr>
<td>ABBR-EXPANSION#</td>
<td>full name</td>
<td>CNN is an acronym for what?</td>
</tr>
<tr>
<td>ANIMAL#</td>
<td>animal</td>
<td>What animal has the biggest eyes?</td>
</tr>
<tr>
<td>CARDINAL#</td>
<td>cardinal number</td>
<td>How many acres in a mile?</td>
</tr>
<tr>
<td>COLOR#</td>
<td>colors</td>
<td>What colors make up a rainbow?</td>
</tr>
<tr>
<td>DATE#</td>
<td>dates</td>
<td>What is Judy Garland's date of birth?</td>
</tr>
<tr>
<td>DEFINITION#</td>
<td>definition of sth.</td>
<td>What are geckos?</td>
</tr>
<tr>
<td>DESCRIPTION#</td>
<td>description of sth.</td>
<td>What do economists do?</td>
</tr>
<tr>
<td>DISEASE#</td>
<td>diseases and medicine</td>
<td>What is a fear of failure?</td>
</tr>
<tr>
<td>EVENT#</td>
<td>events</td>
<td>What events happened January 26, 1978?</td>
</tr>
<tr>
<td>FOOD#</td>
<td>food</td>
<td>What do penguins eat?</td>
</tr>
<tr>
<td>LOCATION#</td>
<td>locations</td>
<td>What are the world's four oceans?</td>
</tr>
<tr>
<td>MANNER#</td>
<td>manner of an action</td>
<td>How can you be happy?</td>
</tr>
<tr>
<td>MEASURE#</td>
<td>measure</td>
<td>How tall is kilimanjaro?</td>
</tr>
<tr>
<td>MONEY#</td>
<td>prices</td>
<td>How much was the minimum wage in 1991?</td>
</tr>
<tr>
<td>ORDINAL#</td>
<td>ranks</td>
<td>What chapter of the Bible has the most verses?</td>
</tr>
<tr>
<td>ORGANIZATION#</td>
<td>organizations</td>
<td>Name Pittsburgh's baseball team.</td>
</tr>
<tr>
<td>OTHERNUMEXP#</td>
<td>other num expressions</td>
<td>What was Einstein's IQ?</td>
</tr>
<tr>
<td>PERCENT#</td>
<td>fractions</td>
<td>What percentage of the body is muscle?</td>
</tr>
<tr>
<td>PERSON#</td>
<td>an individual</td>
<td>What is Alice Cooper's real name?</td>
</tr>
<tr>
<td>PLANT#</td>
<td>plants</td>
<td>What is state tree of Nebraska?</td>
</tr>
<tr>
<td>PROFESSION#</td>
<td>professions</td>
<td>What is Larry King's job?</td>
</tr>
<tr>
<td>REASON#</td>
<td>reasons</td>
<td>Colin Powell is famous for what?</td>
</tr>
<tr>
<td>SPORT#</td>
<td>sports</td>
<td>What sport does Chris Jogis play?</td>
</tr>
<tr>
<td>SUBSTANCE#</td>
<td>elements and substances</td>
<td>What is glass made of?</td>
</tr>
<tr>
<td>TITLE#</td>
<td>title of creative pieces</td>
<td>Jude Law acted in which film?</td>
</tr>
<tr>
<td>OTHER#</td>
<td>all other EATs</td>
<td>A corgi is a kind of what?</td>
</tr>
</tbody>
</table>

Table 1: List of the EATs used in the dataset

Rule based EAT recogniser

rules.lisp All the hand written rules for the EAT recognition.
predicates.lisp The predicates associated with the variables used in the rules.
data-loader.lisp Contains the statements for loading the files containing lists of words acquired from WordNet.

rule-based-eat-PEREZL.tar.gz contains the .lisp files together with a /data/ directory with the .txt files containing the lists of words acquired from WordNet.

Data driven EAT recogniser

Our Java project for generating the ARFF [5] file contains the following classes:

EatDataDrivenParser.java It is the main class of the project, given an input file with the format of the given training data set it generates the ARFF file. The main method in this class is parseInput(String inputFileName, String outputArff) which reads the input file line by line, and processes each of the sentences given in the input file format. Generating the appropriate data training example, according to the features defined (Section 3).

WordNetUtilities.java This class is responsible for building the list of hyponyms for a given concept. It makes use of the JWNL API [1] to access WordNet. Firstly, it calls the method JWNL.initialize() to initialize the application that is making use of it. Then, the method Dictionary.getInstance() is invoked to get an instance of the dictionary installed on the system. The main method used in our WordNetUtilities class is PointerUtils.getInstance().getHyponymTree(word.getSense(sense)).

eatDataDrivenParser9t.jar Contains the java classes and metadata of our project. Generates the ARFF file format with the selected set of best 9 features.

The syntax to generate the ARFF file is the following:

java -jar eatDataDrivenParser9t.jar inputFileName outputFileName

eatDataDrivenParser-PEREZL.tar.gz contains the distribution of our program and all the necessary files for its execution, it contains:
- eatDataDrivenParser9t.jar
- jwnl.jar (WordNet access library)
- WordnetInterface (directory that contains the necessary files to configure the WordNet access library)
- DefinitionSemanticRelatedWords.txt file containing the hand-crafted list of words semantically related to definition questions.
- prepositionsOfLocation.txt file containing the hand-crafted list of prepositions of location.

EatDataDrivenParser-SourceCode-PEREZL.tar.gz contains the .java files