Question classifier for the Question Answering system Hajen

Christopher Käck, Robin Leussier

1 Introduction

This is written as a part of the LTH project course EDAN50: Intelligent systems. Our task was to test and evaluate different techniques for *question classification*. The classifier is supposed to be part of a bigger system called Hajen. Hajen is an artificial intelligence which is designed to compete in the swedish quiz game *Kvitt eller dubbelt*.

The classification of the questions is to be used by the answer extraction part of the system. It will utilize the classes to see where in an interesting paragraph (retrieved by another part of the system) it should look for the possible answers.

2 The dataset

For the dataset, we decided at the beginning of the project to use the questions from the cards of the swedish board game version of *Kvitt eller dubbelt*.

2.1 Cards from *Kvitt eller dubbelt*

We transcribed a set of cards from the game *Kvitt eller dubbelt*. Each card has a number, a general category (such as *Animals and Nature*, *Books and Movies* or *Sport and Leisure*) on the backside and a subcategory or subject (such as *Dog, The Aristocats* or *Hockey*) on the front side with six questions. The six questions each have one of the following points 250, 500, 1000, 2000, 5000 or 10000 indicating their difficulty.

The first stage of the project was to transcribe the cards to a file, in this format:

Number Category Star Subject Points Question Answer AnswerExplanation CoarseType

So we had for example :

*149 Djur och natur * Hunden 250 Vad kallas hundens ungar? Valpar*

These were then transformed into RDF triples.

3 The classes

Our goal was to classify questions it into two types of categories: the *question classes* and the *answer classes*.

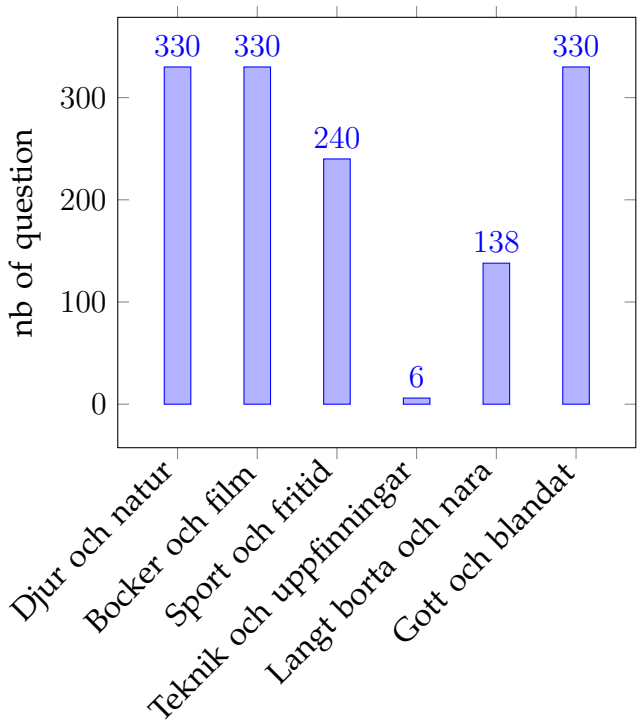
3.1 Question classes

This classification tries to map the question into one of the categories found on the backside of the card. There was 7 possible categories.

- Djur och natur - Animals and Nature
- Böcker och film - Books and movies
- Sport och fritid - Sports and leisure
- Teknik och uppfinningar - Technology and inventions
- Långt borta och nära - Near and far
- Gott och blandat - Mixed
- Musik och underhållning - Music and Entertainment

We are trying to find the topic for the question.

3.1.1 Distribution of the classes



3.2 Answer classes

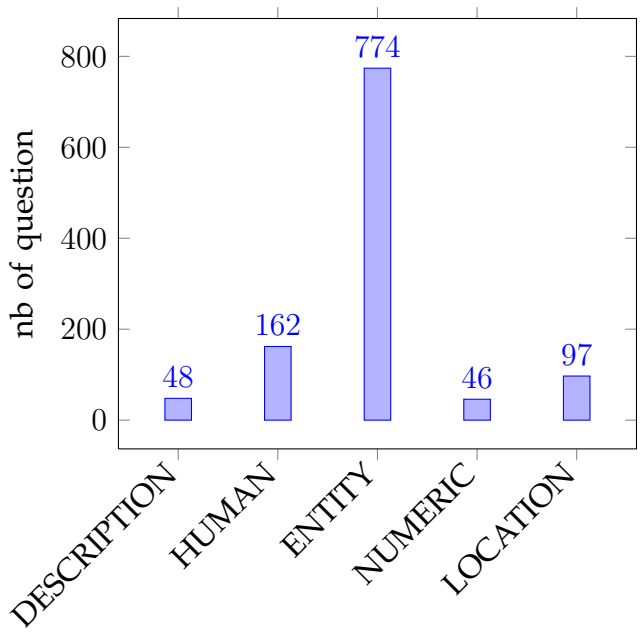
In this classification we are trying find out what the question is asking for. This can be classes like *The question is asking for a DESCRIPTION*. We choose to base our classes on the coarse classes used in the work of Xin Li and Dan Roth [4].

- DESCRIPTION
- HUMAN
- ENTITY
- NUMERIC
- LOCATION

We added two classes to fit the questions from Kvitt eller dubbelt, but have not used them in the tests displayed below.

- ACTION
- BINARY

3.2.1 Distribution of the classes



4 Machine learning algorithms

Machine learning, a part of the artificial intelligence field, is the development, the analysis and the implementation of automated methods that allow a machine to evolve through a learning process. This allows it to handle tasks that are difficult or impossible to handle with more conventional algorithmic means. The first stage of the analysis in supervised machine learning algorithms is the classification, which aims to tag each data by associating it to a class. Many different data mining techniques exists, such as Linear regression, Support Vector Machine and Bayesian reasoning.

4.1 Linear Regression

The linear regression model is the oldest and most popular used predictive model in the field of machine learning. The goal of this model is to predict the values to n data points $(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)$ by a regression model given by

$$y = a_0 + a_1 x_1 + a_2 x_2 + ... + a, X_n$$

where a_0 and a_1 are the constants of the regression model.

4.2 SVM

Support vector machines (SVM) are a group of supervised learning methods that can be applied to classification or regression. It is known as one of the most successful classification algorithms for data mining.[1]

4.2.1 String Kernel

In SVM, the String Kernel is used to transform data from its original space to one where it can be more easily separated and grouped by comparing similarity in substrings of the inputs.

4.2.2 Polykernel

The polynomial kernel is a kernel function, it represents the similarity of vectors (training samples) in a feature space over polynomials of the original variables.

4.2.3 libSVM

LibSVM is an integrated software for support vector classification, regression and distribution estimation. [2]

4.3 BayesNet

Bayesian networks, is a powerful methodology which allows building flexible joint probabilistic models. Those models represents relevant dependencies among a set of variables.

The Bayes networks are directed acyclic graphs whose nodes represent variables, and whose arcs encode conditional independencies between the variables. The graph provides an intuitive description of the dependency model and defines a simple factorization of the joint probability distribution leading to a flexible model which is compatible with the encoded dependencies.[3]

5 Experiments and results

5.1 Evaluation techniques

To evaluate the precision of our classifier we used *ten-folds*. This means that we took the data and split it into 10 parts. Then we took each of these ten parts and used as input to the model we trained on the other nine parts of the input. Then we checked how many percent of the input we managed to classify correctly.

5.2 Pre-processing

All of the machine learning algorithms we used took numeric inputs in the *libsvm* format, except the SVM version called `StringKernel`. The libsvm format looks like this.

```
 1 2:0.4 22:0.1
-1 3:1 2:0.1
```

The first column represents the class that the row belongs to. The other are pairs of features and weights representing a sparse vector. We used *bag of words* as our features, so the feature number represents the presence of a certain word. The features are weighted with *TF-IDF* calculated like this. [5]

$$w = \frac{\sum(WordInQuestion)}{\sum(WordInAllDocuments)}$$

The `StringKernel` took strings and a nominal class as input (in XRFF format[SRC]).

5.3 Results

5.3.1 Question classes

Algorithm	Classied correct
SVM: StringKernel	68.9229 %
SVM: PolyKernel	65.5022 %
SVM: libSVM linear kernel	65.7205 %
BayesNet	65.4294 %
LinearRegression	68.0495 %

Here is the confusion matrix for the `StringKernel`, which shows you what classes the classifier usually makes mistakes on.

```
=== Confusion Matrix ===
  a   b   c   d  e   f <- clas. as
204  11  35  27  0  53 | a = gott
 18  92   6   3  0  19 | b = langt
 47   5 237   9  0  32 | c = djur
 48   6  18 149  0  19 | d = sport
  2   1   0   2  0   1 | e = teknik
 38   3  16   8  0 265 | f = bocker
```

5.3.2 Answer classes

Algorithm	Classied correct
SVM: StringKernel	80.278 %
SVM: PolyKernel	81.0599 %
SVM: libSVM linear kernel	80.9731 %
BayesNet	70.9818 %
LinearRegression	81.2337 %

Here is the confusion matrix for the `StringKernel`

```
=== Confusion Matrix ===
  a   b   c   d   e  <-  clas. as
471  30   3   2   4 | a = entity
 37  86   1   1   0 | b = human
 29   8  32   0   0 | c = location
  7   0   0  19   0 | d = numeric
 48   4   0   0  11 | e = descr.
```

6 Conclusions

The `StringKernel` and `libSVM` perform best generally. The `StringKernel` however is very slow to train compared to `libSVM`.

The better results in the answer classes can partly be explained by the uneven distribution of the different types. Since entity is very over-represented in the training set, the algorithms get very good results by simply giving precedence to the `entity` class. If we only include 200 of the entity tagged questions but keep everything from the other classes (577 questions in total) our percision drops, for `libSVM`, to 67.4177 %.

6.1 Reworking the answer classes

It seems clear to us that the *answer classes* needs more work. Right now they are riddled with ambiguity. Is a description of a human a *HUMAN* or *DESCRIPTION* class? Questions that intuitively belong in the same class wind up in different classes, which makes it hard for both humans and machines to predict the classes. This also applies to the Question classes where *Gott och blandat* is the most usual mistake to make for almost all classes, it doesn't seem to be mutually exclusive.

We have held out working on deciding and reclassifying all the cards, since an important factor is that the classes needs to be helpful to the rest of the system. Thus it must be up to the rest of the system to put requirements what classes it would like to have as input.

We need the classes to be *mutually exclusive*, have *complete coverage* and we need them to be *useful by the other parts of the question answer system*.

6.2 Extending the work

There is many improvement paths to explore. One step that needs to be examined more clearly is the *answer classes* discussed above. These needs to be chosen with much care to maximize the use for the complete system.

Other steps to improve the results of the classifier would be to explore new features. Such as stemmed words, n-grams, part of speech tagging and much more.

There is also work to be done to automate this in a good way, we built a simple web API which is queryable via POST requests, but the classifier needs to be integrated into the complete system pipeline to ensure speed requirements are met.

References

[1] About Support Vector Machines : http://www.support-vector-machines.org/

[2] About the SVM Library : http://www.csie.ntu.edu.tw/~cjlin/libsvm/

[3] About Bayes Network : http://www.meteo.unican.es/research/datamining

[4] About the Question classification : http://cogcomp.cs.illinois.edu/Data/QA/QC/definition.html

[5] R. Baeza-Yates, B. Ribeiro-Net. Modern Infromation Retrieval. Pearson. Second Edition. 2011.