Thursday, February 9, Text-Mining Exercise

Kasper Jensen (kasjens@cbs.dtu.dk)  
Olga Regina (olga@cbs.dtu.dk)

Introduction

Imagine having a research question and no database resource to help you answer this question. This is a common issue in cutting-edge science. Text mining is a tool to help you create your own resource from literature such as PubMed or patient records by reading millions of small pieces of text on an automated manner. The purpose of this exercise is to present some of the basic concepts and challenges of text mining through a hands-on exercise.

Logon to the server

Do the text-mining exercise on the CBS organism server. If you are connected to the Internet through DTU wireless you should first logon to the CBS network using your terminal. If you are using Windows you should use PuTTY as your terminal.

```
ssh stud0**@login.cbs.dtu.dk
```

** Is to be replaced with the number assigned to you.

Once logged on to the CBS network logon to organism server by the following command.

```
ssh organism
```

Setup your working folder

First we will make a workspace for this exercise, for you to work in. Setup a workspace for the text-mining exercise using the following commands.

```
mkdir textmining
cd textmining
```
Create a link to the exercise files from your workspace

`ln -s /home/projects/kasjens/textmining/*`.

**Your dictionaries**

The words, which will eventually form your resource, are included in your dictionaries. The dictionaries contain the words that are to be recognized by text mining and are one of the most critical input components to your resource. A good dictionary usually takes months to construct but for demonstration purposes we have created some simplified dictionaries for you to work with.

In the folder `dataset` in your workspace you will find two sets of dictionaries.

**Food-disease dictionaries:** This dictionary allows you to create a resource linking the consumption of food and the associated disease phenotypes.

- `dataset/food_dictionary.tsv`
- `dataset/omim_disease_dictionary.tsv`

Have a look at the content of the one of the dictionary files by typing:

```bash
head dataset/food_dictionary.tsv
```

and

```bash
head dataset/omim_disease_dictionary.tsv
```

**Smoking – infertility dictionaries:** Allows you to create a resource linking smoking, genes and disease phenotype infertility.

- `dataset/smoking_gene_dictionary.tsv`
- `dataset/smoking_phenotype_dictionary.tsv`
- `dataset/smoking_disease_dictionary.tsv`
Tag your documents

The documents to be “mined” (etc. PubMed or patient records) are tagged; which means that the occurrence of the words and their position is determined and a file with the tagged entities is produced.

The tagging of words is a critical step in text mining and advanced methods apply fuzzy matching. You can find an explanation of fuzzy matching at the wiki-page: http://en.wikipedia.org/wiki/Fuzzy_matching

For demonstration purposes the following tagging will include simple word lookup (no fuzzy matching). The following script will tag the words from your dictionary in the provided records. For the following exercise we will tag the records of PubMed/MEDLINE.


For the command below [your document] is to be replaced below with the file specified for you during the exercise and [your dictionaries] is to be replaced by the dictionaries of your choice from above. The file names are space-separated.

```bash
./tag_documents.pl --documents [your document] --dictionaries [your dictionaries] --columns 5 6 > tagged_documents.tsv
```

eg.

```bash
./tag_documents.pl --documents medline12_food_disease_random10k.tsv --dictionaries dataset/food_dictionary.tsv dataset/omim_disease_dictionary.tsv --columns 5 6 > tagged_documents.tsv &
```

This will take approximately 4-5 minutes and you can follow the process by typing:

```bash
tail -f tagged_documents.tsv
```

Blacklisting words which are too common

After tagging, the next step is to re-evaluate the words in your dictionary by looking at their tagging frequency. The words, which are tagged too frequent, may be too common to be of interest.
An example:

The word ‘nut’, which is the nut we eat, is a valid food name. However, nut is used in a lot of other meanings and thus could be too common to be of interest. The same applies to the word ‘water’, which is also a valid food name. You drink water. However, the word water is used in a lot of scientific publications and leads to a lot of useless tagging. These words are to be removed by blacklisting.

```
./wash_by_blacklist.pl --tagdocument tagged_documents.tsv --blacklist blacklist.tsv --output washed_tagged_documents.tsv
```

This will produce a new tagging with a cleaned vocabulary.

Training the language processor

In the following we will train a classifier on the “bag of words” principle to combine the words into pairs. Several methods for this exist. In the following we are going to train a Bayesian network as a language processor (text classifier). The Bayes classifier is a simple probabilistic classifier, based on Bayes’ theorem with strong (naive) independence assumptions.

A subset of the tagged documents is to be used for training of the classifier. However, a significant amount of training data is required for high performance and to avoid over-fitting.

```
./compile_training.pl --documents [your document] --tagdocument washed_document_tagging.tsv --columns 5 6 --output training.tsv
```

e.g.

```
./compile_training.pl --documents medline12_food_disease_random10k.tsv --columns 5 6 --tagdocument washed_tagged_documents.tsv --output training.tsv
```

The script will ask you to decide on whether tagged relations are true (positive), false (negative) or if the tagged entities are noise. After ~ 100 abstracts you should be able to train your classifier. However, for high performance you may need up ~1000 abstracts classified through this method.
Evaluating the performance

The performance is evaluated by classifying known abstracts into false and positive categories. If the abstract is more likely to be describing a true relation it is considered true, otherwise false. From the number of correctly classified instances, performance accuracy is estimated.

Evaluating the performance is a critical step and several evaluations and tests may be needed to ensure an adequate model. If your training set is too small, over-fitting will make the model remember the sentences instead of the language and this has to be solved through construction of training set of sufficient size.

10-fold cross-validation (http://en.wikipedia.org/wiki/Cross-validation_%28statistics%29) is a common method to evaluate your model’s performance while avoiding/detecting over-fitting of your model. However, since our training set is small we will use the same training and test set, which makes it more prone to over-fitting.

```
./evaluate_performance.pl --training training.tsv --evalout evaluation.tsv &
```

And follow the progress of the training by

```
tail -f evaluation.tsv
```

The first column is the number of words used for training and the last column is the accuracy measure.
Questions

Q1) What is the maximum number of words used for training?

Q2) What is the performance of the model with just 1 word?

Q3) What is the highest performance?

Q4) Does your model over-fit?

Read this wiki-page on Bayes’ theorem: http://en.wikipedia.org/wiki/Bayes%27_theorem

Q5) Would you gain in performance by multiplying the probabilities of your model?

Q6) You have trained a classifier to recognize your abstracts describing medical conditions in patients. The probability of the abstract describing a true medical condition is \( P(\text{abstract}) = 0.79 \). The abstract mentions two medical conditions in each different sentence. The first sentence describes diabetes with a probability is \( P(\text{sentence}) = 0.98 \) and the second condition is stroke with a probability of \( P(\text{sentence}) = 0.67 \).

6A) Is diabetes identified as a true medical condition?

6B) Is stroke identified as a true medical condition?