NLP and IR

Lab02: SVM reranking

Aliaksei Severyn

University of Trento, Italy

April 26, 2012

Plan for the lab

- Quick Recap
- SVM re-ranker
- Modeling QA pairs
  - Simple bag-of-features approach
  - Linear and sequence kernels
- Evaluating results
- Possible ideas for improvement

Quick Recap - goal

- What is our ultimate goal?
  - Build some of QA system components
    - QA answer retrieval
    - QA answer re-ranker
    - No answer identification yet
- As course project you will get to implement some of the other modules:
  - For example, various linguistic annotators

Quick Recap – our data

- We used Answerbag QA collection scraped from the web to build and test our QA models ~ 180k QA pairs
- Fairly high quality, since we consider only "professionally researched" questions
- The link for the data is in the slides of the previous lecture (you will also need for today's lecture)

Quick Recap – retrieval module

- We used Lucene as a search engine to retrieve most relevant answer candidates quiven a query question
- Two main steps:
  - Index
  - Retrieve

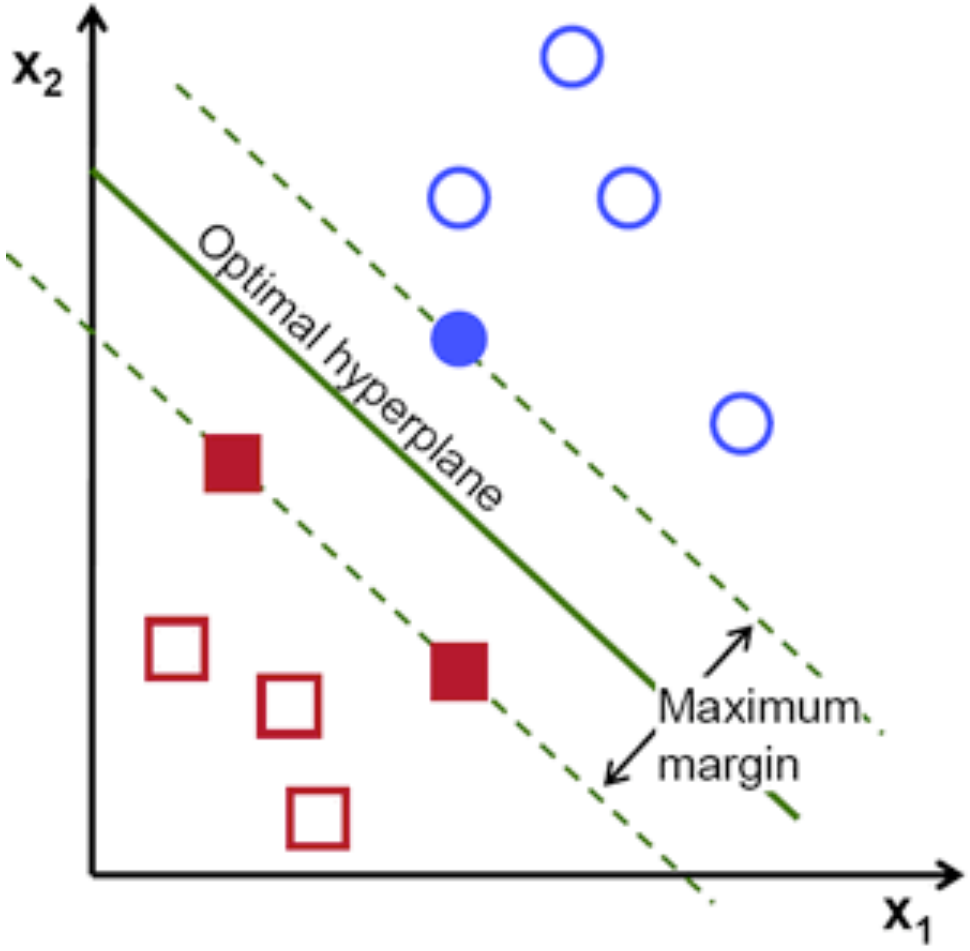
Quick Recap – retrieval module

- This is the very first step
- Uses very simple scoring model, e.g. cosine similarity (by default)
- We used evalSearchEngine.py to evaluate the quality of our retrieval module
- Can try more advanced scoring models

Next steps – SVM reranker

- Use a re-ranker to improve the rankings output by the search engine
- Experiment with several natural language models of QA pairs to capture salient features between questions and answers
- Evaluate if we can improve the rankings of the search engine

Recap on SVMs

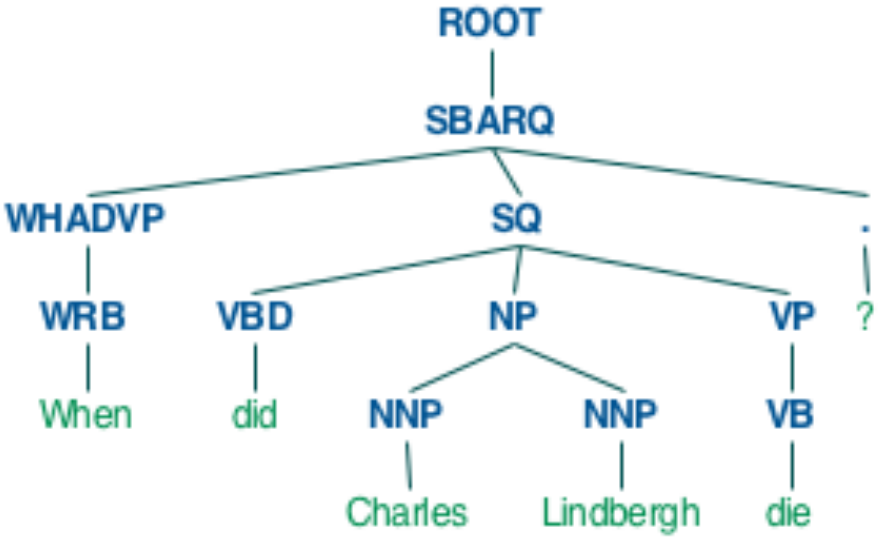


Getting practical with using SVM

- Input format
  - Feature vectors
  - Objects, e.g. sequences, trees
- Learning
  - Classification, regression, re-ranking
  - Kernels: sequence kernels and tree kernels
- Classifying

Question classification on DESC data

1 |BT| (ROOT (SBARQ (WHADVP (WRB When))(SQ (VBD did)(NP (NNP Charles)(NNP Lindbergh))(VP (VB die)))(. ?))) |ET|



Let's practice

Train an SVM model to classify questions

```
$ ./svm_learn -t 5 demo-data/NUM_train.dat model
```

Now test our model

```
$ ./svm_classify  demo-data/NUM_test.dat model
```

Various input formats are possible

<label> |BV| <feature vector> |EV|

More complex

<label> |BT| (tree) |ET| <feature vector> |EV|

SVM for reranking

- Instead of classifying points, we want to learn a classifier on QA pairs
  - Correct QA pair is scored higher than incorrect
  - Provided prediction scores – reorder the rankings of the search engine

Reranking: bag of words features

- The simplest model
  - Bag of words + the similarity score from the search engine
- Important to show how to setup the data for the reranker
- How to train and classify the data
- Evaluate the results

Download the materials

Download the materials for the lab01 and lab02 from the course web site

The step by step instructions are in the README.txt

Open the archive

```
$ tar xvfz lab02.tar.gz
$ cd lab02
```

Build SVM

go under SVM directory

```
$ cd SVM-Light-1.5-rer/
```

type make to build the code

```
$ make
```

go back to the previous directory

```
$ cd ..
```

Generating QA pairs

Generate QA pairs for training:

```
$ python generate_reranking_pairs.py
  questions.5k.txt answers.txt
  results.train.15k
```

Generate QA pairs for testing:

```
$ python generate_reranking_pairs.py -m
  test questions.5k.txt answers.txt
  results.test.15k
```

Looking at the SVM examples file

```
+1      |BT| (BOX (the)(cell)(phone)(used)(tony)
   (stark)(the)(movie)(iron)(man)(was)(vx9400)
   (slider)(phone)(which)(was)(just)(one)(the)
   (mobile)(phones)(used)(the)(movie.)) |BT| (BOX
   (the)(average)(person)(cannot)(trace)(prepaid)
   (cell)(phone)(however)(the)(federal)(government)
   (and)(police)(force)(have)(this)(capability.)
   (while)(they)(cannot)(determine)(person)(exact)
   (location)(they)(can)(find)(what)(cell)(phone)
   (towers)(are)(being)(used)(and)(use)(this)
   (information)(trace)(the)(phone.)) |ET|
   1:2.28489184 |BV| 1:0.65760440 |EV|
```

Training SVM

Learn

```
$ ./SVM-Light-1.5-rer/svm_learn -t 5 -F 2 -C + -W
  R -V R -S 0 -N 1 svm.train model
```

Classify

```
$ ./SVM-Light-1.5-rer/svm_classify  svm.test model
  pred
```

Training and classification should take a few minutes (depending on your machine)

Comments on SVM options

-F 2 – linear kernel on the bag of words

-C + - combine contribution from trees and vectors

-W R - apply reranking on trees

-V R - apply reranking on vectors

-S 0 - linear kernel on the feature vector;

-N 1 - no normalization on the feature vector;

Evaluate the results

Run the evaluation script

```
$ python evReranker.py svm.test.res pred
```

The file `svm.test.res` contains the output of the search engine and the golden standard (correct QA pairs), which we use to evaluate how well we did w.r.t. search engine

We also use the file `pred` containing SVM predictions to rerank the output of the search engine

We improve only slightly

	IR	SVM
MRR :	72.48	72.71
REC-1@01:	73.44	73.67
REC-1@02:	84.34	85.47
REC-1@03:	89.33	89.44
REC-1@04:	90.92	91.37
REC-1@05:	93.64	92.96
REC-1@06:	95.23	94.32
REC-1@07:	96.25	95.35
REC-1@08:	97.28	96.59
REC-1@09:	97.50	97.39
REC-1@10:	98.18	98.07

MRR - mean reciprocal rank (http://en.wikipedia.org/wiki/Mean_reciprocal_rank)

REC-1 - percentage of questions with at least 1 correct answer in the top @X positions (useful for tasks were questions have at most one correct answer)