NLP and IR

Lab04: SVM reranking

Aliaksei Severyn

University of Trento, Italy

May 9, 2013

Plan for the lab

- Quick Recap on Supervised Learning
- SVM re-ranker (for QA)
  - pointwise
  - pairwise
- Modeling QA pairs
  - Feature model
  - Structural model with kernels
- Evaluating results

Recap on supervised learning

Given a labeled training set:

$$\{(x_i, y_i)\}_{i=1}^{N} \in \mathcal{X} \times \mathcal{Y}$$

In supervised learning the goal is to learn a decision function:

$$h : \mathcal{X} \to \mathcal{Y}$$

That minimizes the empirical loss:

$$L = \frac{1}{N} \sum_{i=1}^{N} \Delta(h(x_i), y_i)$$

Learning tasks

Binary classification:

$$\mathcal{Y} \in \{0, 1\}$$

Multi-class classification:

$$\mathcal{Y} \in \{0, .., K\}$$

Reranking:

$$\mathcal{Y} \in \{1, .., R\}$$

Output is related on ordinal scale

Example reranking output

Candidates	Prediction	True Rank
DOC1	4	3
DOC2	5	5
DOC3	1	1
DOC4	2	2
DOC5	3	4

Example reranking output in QA

Candidates	Prediction	True Rank
DOC1	4	1
DOC2	5	0
DOC3	1	1
DOC4	2	0
DOC5	3	0

Reranking with SVM

SVM is inherently a **binary classifier**

How can we solve a **reranking task** using binary SVMs?

Reranking with SVM

SVM is inherently a **binary classifier**

How can we solve a **reranking task** using binary SVMs?

A naïve approach:

- train a binary classifier to classify documents as relevant or not
- obtain the final ranking by sorting on the prediction score

Reranking with SVM

SVM is inherently a **binary classifier**

How can we solve a **reranking task** using binary SVMs?

We already found a clever way to do this for multi-class classification, i.e. one-vs-all

Use the **reduction approach**!!!

Transforming into binary classification

Candidates	Prediction	True Rank
DOC1	4	3
DOC2	5	5
DOC3	1	1
DOC4	2	2
DOC5	3	4

h(DOC3) > h(DOC4)

h(DOC3) > h(DOC1)

h(DOC3) > h(DOC5)

h(DOC3) > h(DOC2)

h(DOC4) > h(DOC1)

h(DOC4) > h(DOC5)

h(DOC4) > h(DOC2)

h(DOC1) > h(DOC5)

h(DOC1) > h(DOC4)

h(DOC5) > h(DOC2)

Formal problem transformation

Binary

$$\min_{\mathbf{w},\xi_i\geq 0} \quad \frac{1}{2}\mathbf{w}^T\mathbf{w} + \frac{C}{n}\sum_{i=1}^{n}\xi_i$$

$$s.t. \quad \forall i \in \{1, \ldots, n\}: \; y_i(\mathbf{w}^T\mathbf{x}_i) \geq 1 - \xi_i$$

Reranking (ordinal regression)

$$\min_{\mathbf{w},\xi_{ij}\geq 0} \quad \frac{1}{2}\mathbf{w}^T\mathbf{w} + \frac{C}{m}\sum_{(i,j)\in\mathcal{P}}\xi_{ij}$$

$$s.t. \quad \forall (i,j) \in \mathcal{P}: \; (\mathbf{w}^T\mathbf{x}_i) \geq (\mathbf{w}^T\mathbf{x}_j) + 1 - \xi_{ij}$$

$$\mathcal{P} = \{(i,j) : y_i > y_j\}$$

$$h(\mathbf{x}_i) > h(\mathbf{x}_j) \iff y_i > y_j.$$

Reranking for QA

Task:

- Improve upon search engine
- Learn a model to produce a better ranking of the answer candidates

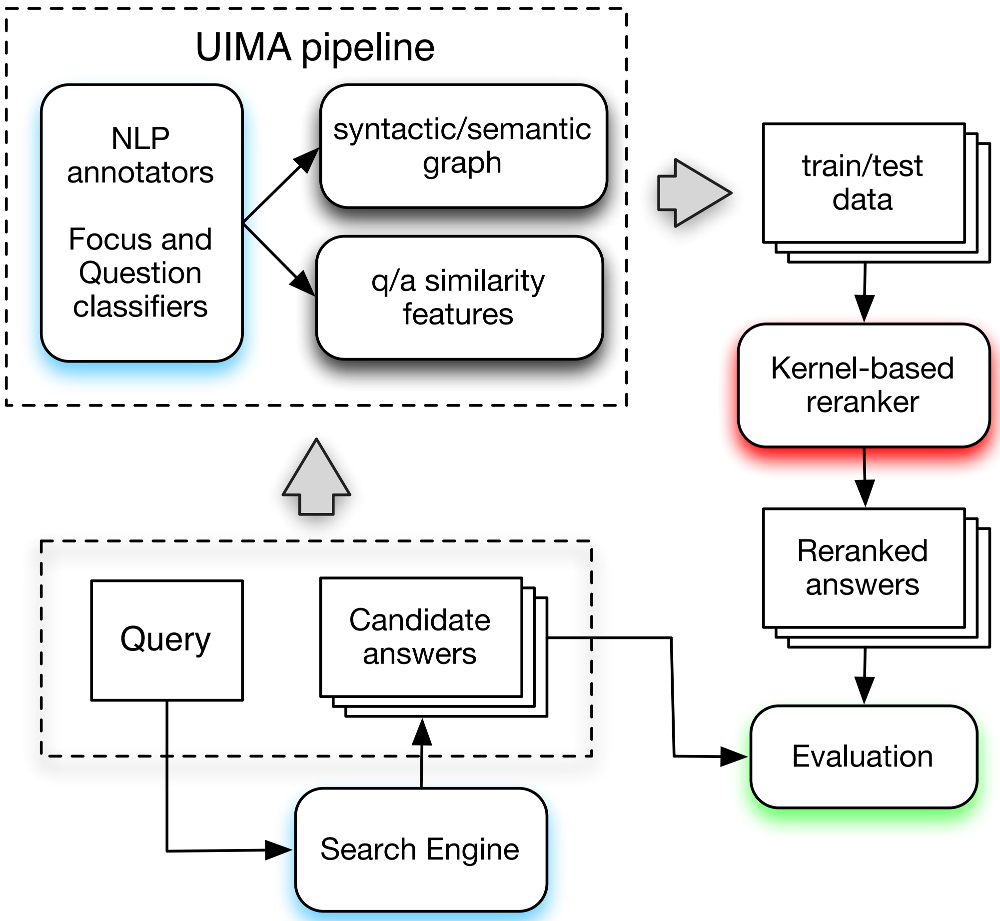
Approach:

- Naïve classifier (pointwise)
  - Can use any off-the-shelf learning method, lower performance, suffers from class imbalance
- Pairwise reranker
  - Better accuracy, slower (depends on the size of the candidate set)
- Listwise (SVM-rank)
  - Better accuracy, faster, can optimize for global metrics, e.g. MAP

Representations:

- Pairwise similarity features
- Structural representation

Architecture of a QA pipeline



QA system components

- Components of a typical QA system
  - QA answer passage retrieval (Lab 01)
    - High efficiency & high recall
  - QA answer passage scoring (today)
    - Lower efficiency & high accuracy
  - Answer phrase identification (out of scope)
    - Greatly relies on the upstream components

TREC QA dataset

A set of "de facto" QA datasets used in research to test QA systems

Provides:

- Specifies supporting corpora to search for answer candidates (requires an LDC licence)
  - answer patterns
- A set of retrieved answers
  - provides relevancy judgements

More info available:

http://trec.nist.gov/data/qa.html

TREC QA dataset

- TREC-8 (1999) QA Data
- TREC-9 (2000) QA Data
- TREC 2001 QA Data
- **TREC 2002 QA Data**
- **TREC 2003 QA Data**
- TREC 2004 QA Data

TREC QA examples

Mainly focus on **factoid questions**, i.e. ask about who, what, when, where, etc.

- In what country did the game of croquet originate?
- Who is Tom Cruise married to?
- What year was Alaska purchased?

Simpler than **non-factoid questions** (Answerbag):

- How do I cook pasta?
- How do i test a pcv valve?

Representations of QA pairs

Pairwise similarity features (**next lab**)

- Encode how similar are a given question and its candidate answer
- Similarity is represented as a **single score**
- Many various similarity metrics can be used to capture different **aspects of similarity**, e.g. lexical, syntactic, semantic, etc.
- May require **significant** engineering effort

Structural representations

- Input objects, e.g. sequences, trees, graphs, can be **treated directly**
- Kernel functions allow for **automatic feature engineering**
- Can result in **higher accuracy**
- **Slow** to train

Using SVM-TK for reranking

Input format:

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

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

Parameter options:

-t 5 advanced kernels

-F 1 type of the kernel applied to structures (STK)

-W R reranking on tree structures

-V R reranking on feature vectors

-C + cominbation of tree and feature vectors

Similarity feature vector

Input format:

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

Parameters:

-t 5 –C V –V R

Structural representation

Each example encodes a tuple of QA pairs

<(Q, A1), (Q, A2)>

Label is +1 if rank((Q, A1)) > rank((Q, A2)) and -1 otherwise

Input format:

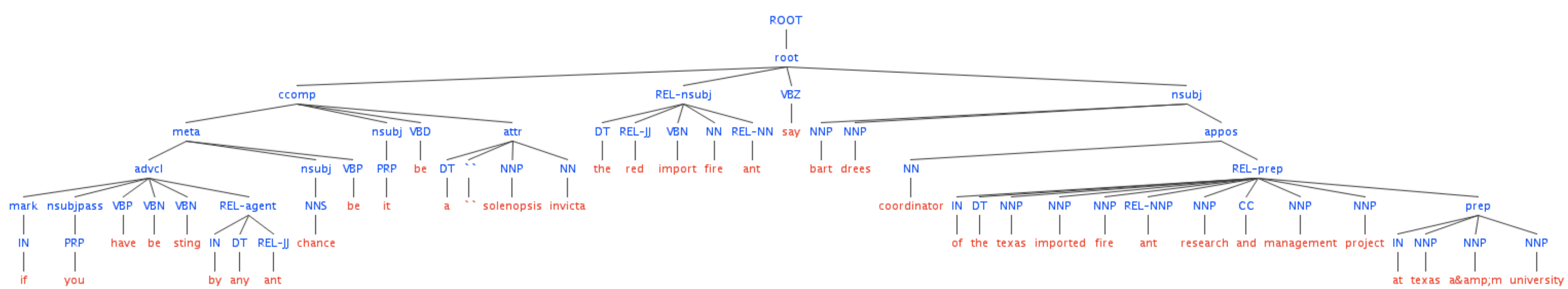
<label> |BT| Q |BT| A1 |BT| Q |BT| A2|ET| <fvec1> |BV| <fvec2> |EV|

Parameters:

-t 5 –F 3 –C + -W R –V R

Structural representations

- Encode structural patterns of relating correct question/answer pairs
- Rely on tree kernels to automatically extract all kinds of features



Pointwise vs. Pairwise vs. Listwise

Pointwise

	IR	SVM	(abs)	(rel)
MRR:	28.02 ± 2.94	32.65 ± 1.65	+4.63% ± 2.38	+17.30% ± 10.70
MAP:	0.22 ± 0.02	0.25 ± 0.01	+0.03% ± 0.02	+13.67% ± 10.42
P@1:	18.17 ± 3.79	21.34 ± 1.29	+3.17% ± 2.64	+20.92% ± 21.74

Pairwise

	IR	SVM	(abs)	(rel)
MRR:	28.02 ± 2.94	36.09 ± 1.01	+8.07% ± 3.48	+30.08% ± 15.60
MAP:	0.22 ± 0.02	0.27 ± 0.01	+0.06% ± 0.02	+25.87% ± 11.02
P@1:	18.17 ± 3.79	26.34 ± 0.51	+8.17% ± 4.10	+51.03% ± 37.08

Listwise

	IR	SVM	(abs)	(rel)
MRR:	28.02 ± 2.94	36.23 ± 1.32	+8.21% ± 3.17	+30.42% ± 14.00
MAP:	0.22 ± 0.02	0.28 ± 0.01	+0.07% ± 0.01	+30.52% ± 8.58
P@1:	18.17 ± 3.79	25.37 ± 0.70	+7.20% ± 3.59	+44.74% ± 30.87

Features vs. Structural (pairwise)

Similarity Features

	IR	SVM	(abs)	(rel)
MRR:	28.02 ± 2.94	36.09 ± 1.01	+8.07% ± 3.48	+30.08% ± 15.60
MAP:	0.22 ± 0.02	0.27 ± 0.01	+0.06% ± 0.02	+25.87% ± 11.02
P@1:	18.17 ± 3.79	26.34 ± 0.51	+8.17% ± 4.10	+51.03% ± 37.08

Structural

	IR	SVM	(abs)	(rel)
MRR:	28.02 ± 2.94	38.91 ± 2.20	+10.89% ± 3.77	+40.10% ± 15.96
MAP:	0.22 ± 0.02	0.31 ± 0.02	+0.09% ± 0.03	+40.85% ± 14.25
P@1:	18.17 ± 3.79	**28.17** ± 2.74	+10.00% ± 4.73	+60.93% ± 36.70

In class experiment

```
# go to the root folder of the class source repo
cd your_path_to_NLPIR

# Uncomment the PARAMS variable in
scripts/trec_train_test_eval.sh
PARAMS="-t 5 -C V -j 5”
# Test the pointwise (classifier) approach
sh scripts/trec_train_test_eval.sh data/trec.qa/
  poschunk.pointwise/fold0
# Again change PARAMS to enable pairwise approach:
PARAMS="-t 5 -C V –V R”
# Test the pairwise approach
sh scripts/trec_train_test_eval.sh data/trec.qa/
  poschunk.pairwise/fold0
```

Results

Pointwise

	IR	SVM
MRR:	24.01	31.83
MAP:	0.19	0.26

	IR	SVM
REC-1@01:	12.80	19.51

Pairwise

	IR	SVM
MRR:	24.01	36.30
MAP:	0.19	0.27

	IR	SVM
REC-1@01:	12.80	26.83

Summary

Reranking with SVM

- Problem reduction to binary SVM

Experiments on TREC QA

- Pointwise
- Pairwise
- Listwise

Representations

- Similarity features
- Structural representation