Advanced Natural Language Processing and Information Retrieval

LAB3: Kernel Methods for Reranking

Alessandro Moschitti
Department of Computer Science and Information Engineering
University of Trento
Email: moschitti@disi.unitn.it
Preference Reranking
slides at:
http://disi.unitn.it/moschitti/teaching.html

LAB3: Ranking with Tree Kernels
Download:
LAB3.zip
The Ranking SVM

[Herbrich et al. 1999, 2000; Joachims et al. 2002]

- The aim is to classify instance pairs as correctly ranked or incorrectly ranked
  - This turns an ordinal regression problem back into a binary classification problem
- We want a ranking function \( f \) such that
  \[ x_i > x_j \iff f(x_i) > f(x_j) \]
- ... or at least one that tries to do this with minimal error
- Suppose that \( f \) is a linear function
  \[ f(x_i) = w \cdot x_i \]
The Ranking SVM

- Ranking Model: $f(x_i)$
The Ranking SVM

- Then (combining the two equations on the last slide):

\[ x_i > x_j \text{ iff } w \cdot (x_i - x_j) > 0 \]

- Let us then create a new instance space from such pairs:

\[ z_k = x_i - x_k \]

\[ y_k = +1, -1 \text{ as } x_i \geq x_k \]
Support Vector Ranking

\[
\begin{aligned}
\min & \quad \frac{1}{2} \|\vec{w}\| + C \sum_{i=1}^{m} \xi_i^2 \\
\text{s.t.} & \quad y_k (\vec{w} \cdot (\vec{x}_i - \vec{x}_j) + b) \geq 1 - \xi_k, \quad \forall i, j = 1, \ldots, m \\
& \quad \xi_k \geq 0, \quad k = 1, \ldots, m^2
\end{aligned}
\]

\[y_k = 1 \text{ if } rank(\vec{x}_i) > rank(\vec{x}_j), -1 \text{ otherwise, where } k = i \times m + j\]

- Given two examples we build one example \((x_i, x_j)\)
The local model is a system providing the initial rank.

Preference reranking is superior to ranking with an instance classifier since it compares pairs of hypotheses.
More formally

- Build a set of hypotheses: Q and A pairs
- These are used to build pairs of pairs \( \langle H^i, H^j \rangle \)
  - positive instances if \( H^i \) is correct and \( H^j \) is not correct
- A binary classifier decides if \( H^i \) is more probable than \( H^j \)
- Each candidate annotation \( H^i \) is described by a structural representation
- This way kernels can exploit all dependencies between features and labels
Preference Reranking Kernel

If $H_1 > H_2$ and $H_3 > H_4$ then consider training vectors:

$\tilde{Z}_1 = \phi(H_1) - \phi(H_2)$ and $\tilde{Z}_2 = \phi(H_3) - \phi(H_4) \Rightarrow$ the dot product is:

$\tilde{Z}_1 \cdot \tilde{Z}_2 = (\phi(H_1) - \phi(H_2)) \cdot (\phi(H_3) - \phi(H_4)) =
\phi(H_1) \cdot \phi(H_3) - \phi(H_1) \cdot \phi(H_4) - \phi(H_2) \cdot \phi(H_3) + \phi(H_2) \cdot \phi(H_4)$

$= K(H_1, H_3) - K(H_1, H_4) - K(H_2, H_3) + K(H_2, H_4)$

Let $H_i = \langle q_i, a_i \rangle$, $H_j = \langle q_j, a_j \rangle$

$K(H_i, H_j) = PTK(q_i, q_j) + PTK(a_i, a_j)$
An example of Jeopardy! Question
The abbey is where all English monarchs have been crowned since William the Conqueror.
Adding Relational Links
Links can be encoded marking tree nodes

Methodology:
1- Applying lemmatization or stemming to the leaves
2- Mark (with @ symbol) pre-terminal nodes and higher level nodes if the subtrees are shared in Q and A
3- Ignore stop words in the matching procedure
Representation Issues

- Very large sentences
- The Jeopardy! cues can be constituted by more than one sentence
- The answer is typically composed by several sentences
- Too large structures cause inaccuracies in the kernel similarity and the learning algorithm looses some of its power
Question: Is movie theater popcorn vegan?

Answer:

(01) Any movie theater popcorn that includes butter -- and therefore dairy products -- is not vegan.

(02) However, the popcorn kernels alone can be considered vegan if popped using canola, coconut or other plant oils which some theaters offer as an alternative to standard popcorn.
Shallow models for Reranking: [Severyn & Moschitti, SIGIR 2012]

Question:

is movie theater popcorn vegan

Answer:

any movie theater popcorn that includes butter and therefore dairy products is not vegan

(is) (movie) (theater) (popcorn) (vegan)

(VBZ) (NN) (NN) (JJ) (NN)
Lexical matching is on word lemmas (using WordNet lemmatizer).
Lexical matching is on word lemmas (using WordNet lemmatizer).
Linking question and its answer passages using a relational tag

Marking pos tags of the aligned words by a relational tag: “REL”
Let’s start the LAB3: Ranking with Tree Kernels
SVM-light-TK and Ranking Data

- SVM-light-TK encodes STK, PTK and combination kernels in SVM-light [Joachims, 1999]
- http://disi.unitn.it/moschitti/teaching.html
  - Academic Year: 2015-2016
  - Download: LAB3.zip
Compile the package

- 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 examples for reranking

- **questions.5k.txt**, contains a set of questions
  - each line contains a unique id and the question itself separated by a tab, i.e., "\t"
- **answers.txt** -- contains a set of answers
  - each line contains a unique id and the answer passage itself separated by a tab, i.e. "\t"
Training and testing files

- **results.*.15k**, a rank list for 1,000 questions (contained in questions.5k.txt)
  1. the id of the question
  2. the id of the passage, and
  3. its score from the search engine

- **results.train.15k, results.test.15k**
  - 1000 questions
  - 15 retrieved passages for each question
  - *(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.))*
Building the reranker files

- Generate training examples for reranking
  - `python generate_reranking_pairs.py questions.5k.txt answers.txt results.train.15k`
  - `python2.7`

- Generate testing examples for reranking
  - `python generate_reranking_pairs.py -m test questions.5k.txt answers.txt results.test.15k`
Retrieval results for a question

- 2500744 What kind of cell phone was used in the movie "Iron Man"?
- 2500744 The cell phone used by Tony Stark in the movie "Iron Man" was a LG VX9400 slider phone, which was just one of the LG mobile phones used in the movie.
- 2259459 The average person cannot trace a prepaid cell phone; however, the federal government and police force do have this capability. While they cannot determine a person's exact location, they can find what cell phone towers are being used and use this information to trace the phone.
Generated learning files: svm.train

- +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|


Generated test files: svm.test

+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| EMPTY |ET| 1:2.28489184 |BV| EMPTY |EV|
Training and testing the reranker

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

- SVM-TK options:
  -F 2, tree kernel using words (bag of word)
  -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, normalization on tree, not on the feature vector
Testing and evaluating the reranker

- ./SVM-Light-1.5-rer/svm_classify svm.test model pred
- python evReranker.py svm.test.res pred