Semi-supervised Question Retrieval with Gated Convolutions

Tao Lei

joint work with Hrishikesh Joshi, Regina Barzilay, Tommi Jaakkola, Kateryna Tymoshenko, Alessandro Moschitti and Lluís Màrquez

NAACL 2016
Our Task

Find similar questions given the user’s input question

Application to find duplicate MP3s [duplicate] title

Possible Duplicate:
How can I find duplicate songs?

I'm looking for a program to find duplicate MP3 files.
The program shouldn't use MD5 hashes but it should find similar file names. (Something like Anti-Twin for Windows).

Any help is appreciated.

question from Stack Exchange AskUbuntu
Our Task

Find similar questions given the user’s input question

question from Stack Exchange AskUbuntu

Our goal: automate this process as a solution for QA
Challenges

• Multi-sentence text contains irrelevant details

Title: How can I boot Ubuntu from a USB?

Body: I bought a Compaq pc with Windows 8 a few months ago and now I want to install Ubuntu but still keep Windows 8. I tried Webi but when my pc restarts it read ERROR 0x000007b. I know that Windows 8 has a thing about not letting you have Ubuntu …

Title: When I want to install Ubuntu on my laptop I’ll have to erase all my data. “Alonge side windows” doesnt appear

Body: I want to install Ubuntu from a Usb drive. It says I have to erase all my data but I want to install it along side Windows 8. The “Install alongside windows” option doesn’t appear …

• Forum user annotation is limited and noisy (more on this later)
Solution

(1) a model to better represent the question text
(2) semi-supervised training to leverage raw text data
Model

Model Architecture*:

Choice of encoder:

LSTM, GRU, CNN ... or:

\[
\begin{align*}
\mathbf{c}_t^{(3)} &= \lambda_t \odot \mathbf{c}_t^{(2)} + (1 - \lambda_t) \odot (\mathbf{c}_{t-1}^{(2)} + \mathbf{W}_3 \mathbf{x}_t) \\
\mathbf{c}_t^{(2)} &= \lambda_t \odot \mathbf{c}_t^{(2)} + (1 - \lambda_t) \odot (\mathbf{c}_{t-1}^{(1)} + \mathbf{W}_2 \mathbf{x}_t) \\
\mathbf{c}_t^{(1)} &= \lambda_t \odot \mathbf{c}_t^{(1)} + (1 - \lambda_t) \odot (\mathbf{W}_1 \mathbf{x}_t) \\
\mathbf{h}_t &= \tanh(\mathbf{c}_t^{(3)} + \mathbf{b})
\end{align*}
\]

Why this encoder (or equations)? How to understand it?

*Other architectures possible: (Feng et. al. 2015), (Tan et. al. 2015) etc.
Sentence: “the movie is not that good”

Bag of words, TF-IDF

Neural Bag-of-words (average embedding)
Sentence: “the movie is not that good”

Ngram Kernel (N=2)

not that
the movie
that good
is not
movie is

CNNs

Neural methods as a dimension-reduction of traditional methods
**Sentence:**  
“the movie is not that good”

**String Kernel**

```
the movie  not _ good
movie _ not  is _ that
not _ good  is not ...
```

- The movie  
- not _ good  
- movie _ not  
- is _ that  
- not _ good  
- is not ...

\[
\begin{bmatrix}
0 \\
\lambda^0 \\
\lambda^2 \\
\vdots \\
\lambda^1 \\
0
\end{bmatrix}
\]

- the movie  
- is _ _ good  
- not _ good  

**Neural model inspired by this kernel method?**

- Bigger feature space  
- \( \lambda \in (0, 1) \) penalize skips
“string” convolution

the movie is not that good
“string” convolution

the movie is not that good
“string” convolution

the movie

is not that good
Formulas in the case of 3gram

\[
\begin{align*}
c_t^{(3)} &= \lambda \cdot c_{t-1}^{(3)} + (1 - \lambda) \cdot (c_{t-1}^{(2)} + W_3 x_t) \\
c_t^{(2)} &= \lambda \cdot c_{t-1}^{(2)} + (1 - \lambda) \cdot (c_{t-1}^{(1)} + W_2 x_t) \\
c_t^{(1)} &= \lambda \cdot c_{t-1}^{(1)} + (1 - \lambda) \cdot (W_1 x_t) \\
h_t &= \tanh(c_t^{(3)} + b)
\end{align*}
\]
Formulas in the case of 3gram

\[
\begin{align*}
\mathbf{c}_t^{(3)} &= \lambda \cdot \mathbf{c}_{t-1}^{(3)} + (1 - \lambda) \cdot (\mathbf{c}_{t-1}^{(2)} + \mathbf{W}_3 \mathbf{x}_t) \\
\mathbf{c}_t^{(2)} &= \lambda \cdot \mathbf{c}_{t-1}^{(2)} + (1 - \lambda) \cdot (\mathbf{c}_{t-1}^{(1)} + \mathbf{W}_2 \mathbf{x}_t) \\
\mathbf{c}_t^{(1)} &= \lambda \cdot \mathbf{c}_{t-1}^{(1)} + (1 - \lambda) \cdot (\mathbf{W}_1 \mathbf{x}_t) \\
\mathbf{h}_t &= \tanh(\mathbf{c}_t^{(3)} + \mathbf{b})
\end{align*}
\]

penalize skip grams
weighted average of 1grams (to 3grams) up to position \( t \)
Formulas

\[ c^{(3)}_t = \lambda \cdot c^{(3)}_{t-1} + (1 - \lambda) \cdot (c^{(2)}_{t-1} + W_3x_t) \]
\[ c^{(2)}_t = \lambda \cdot c^{(2)}_{t-1} + (1 - \lambda) \cdot (c^{(1)}_{t-1} + W_2x_t) \]
\[ c^{(1)}_t = \lambda \cdot c^{(1)}_{t-1} + (1 - \lambda) \cdot (W_1x_t) \]
\[ h_t = \tanh(c^{(3)}_t + b) \]

\[ \lambda = 0 : \quad c^{(3)}_t = W_1x_{t-2} + W_2x_{t-1} + W_3x_t \quad \text{(one-layer CNN)} \]
Gated version

\[
\begin{align*}
c_t^{(3)} &= \lambda_t \odot c_{t-1}^{(2)} + (1 - \lambda_t) \odot (c_{t-1}^{(2)} + W_3 x_t) \\
c_t^{(2)} &= \lambda_t \odot c_{t-1}^{(2)} + (1 - \lambda_t) \odot (c_{t-1}^{(1)} + W_2 x_t) \\
c_t^{(1)} &= \lambda_t \odot c_{t-1}^{(1)} + (1 - \lambda_t) \odot (W_1 x_t) \\
h_t &= \tanh(c_t^{(3)} + b) \\
\lambda_t &= \sigma(W x_t + U h_{t-1} + b')
\end{align*}
\]

adaptive decay controlled by gate
Training

- Amount of annotation is scarce

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td># of unique questions</td>
<td>167,765</td>
</tr>
<tr>
<td># of marked questions</td>
<td>12,584</td>
</tr>
<tr>
<td># of marked pairs</td>
<td>16,391</td>
</tr>
</tbody>
</table>

Forum users only identify a few similar pairs
Only 10% of the number unique questions

Ideally, want to use all questions available
Pre-training Encoder-Decoder Network

Encoder trained to pull out important (summarized) information

pre-training recently applied to classification task

- Semi-supervised Sequence Learning. Dai and Le. 2015
Evaluation Set-up

Dataset: AskUbuntu 2014 dump
pre-train on 167k, fine-tune on 16k
evaluate using 8k pairs (50/50 split for dev/test)

Baselines: TF-IDF, BM25 and SVM reranker
CNNs, LSTMs and GRUs

Grid-search: learning rate, dropout, pooling, filter size,
pre-training, ...

5 independent runs for each config.

> 500 runs in total
## Overall Results

<table>
<thead>
<tr>
<th>Model</th>
<th>MAP</th>
<th>MRR</th>
</tr>
</thead>
<tbody>
<tr>
<td>BM25</td>
<td>68.0</td>
<td>62.3</td>
</tr>
<tr>
<td>LSTM</td>
<td>70.1</td>
<td>66.8</td>
</tr>
<tr>
<td>CNN</td>
<td>71.4</td>
<td>71.3</td>
</tr>
<tr>
<td>GRU</td>
<td>71.3</td>
<td>71.3</td>
</tr>
<tr>
<td>Ours</td>
<td>75.6</td>
<td>75.6</td>
</tr>
</tbody>
</table>

Our improvement is significant
Analysis

- **MAP**
  - Full model: 62.3
  - W/o pretraining: 60.7
  - W/o body: 58.2

- **MRR**
  - Full model: 75.6
  - W/o pretraining: 72.9
  - W/o body: 59.1

- **P@1**
  - Full model: 70.7
  - W/o pretraining: 56.6
Pre-training

MRRs quite different

PPLs are close

MRR on the dev set versus Perplexity on a heldout corpus
Decay Factor (Neural Gate)

\[ c_t^{(3)} = \lambda \odot c_{t-1}^{(3)} + (1 - \lambda) \odot \left( c_{t-1}^{(2)} + W_3 x_t \right) \]

Analyze the weight vector over time
Case Study (using a scalar decay)

(a) how can i add guake terminal to the start-up applications

(f) can anyone tell me how to make guake terminal be part of the start-up applications
Case Study (using a scalar decay)

(b) banshee crashes with `\` an unhandled exception was thrown : "

Diagram showing the sequence of events.
Case Study (using a scalar decay)

(c) i get the error message ``` requires installation of untrusted packages every time i try to update after entering my password ...

(d) i recently bought samsung laptop and i facing hard time to boot my pen driver so that i can use ubuntu ...
Conclusions

• AskUbuntu data as a natural benchmark for retrieval and summarization tasks

• Neural model with good intuition and understanding (e.g. attention) can potentially lead to good performance

https://github.com/taolei87/askubuntu

https://github.com/taolei87/rcnn
and ensured each model has a comparable number of instances but this did not result in any improvement. For the baseline, we used the default SVM-Overflow) and a large Wikipedia corpus. The word vectors are fixed to avoid over-fitting across all experiments.

We ran word2vec (Mikolov et al., 2013) to obtain 200-dimensional word embeddings using all Stack Exchange data (excluding Stack Overflow) and a large Wikipedia corpus. The word vectors are fixed to avoid over-fitting across all experiments.

### Table 2: Comparative results of all methods on the question similarity task.

<table>
<thead>
<tr>
<th>Method</th>
<th>Pooling</th>
<th>Dev MAP</th>
<th>Dev MRR</th>
<th>Dev P@1</th>
<th>Dev P@5</th>
<th>Test MAP</th>
<th>Test MRR</th>
<th>Test P@1</th>
<th>Test P@5</th>
</tr>
</thead>
<tbody>
<tr>
<td>BM25</td>
<td>-</td>
<td>52.0</td>
<td>66.0</td>
<td>51.9</td>
<td>42.1</td>
<td>56.0</td>
<td>68.0</td>
<td>53.8</td>
<td>42.5</td>
</tr>
<tr>
<td>TF-IDF</td>
<td>-</td>
<td>54.1</td>
<td>68.2</td>
<td>55.6</td>
<td>45.1</td>
<td>53.2</td>
<td>67.1</td>
<td>53.8</td>
<td>39.7</td>
</tr>
<tr>
<td>SVM</td>
<td>-</td>
<td>53.5</td>
<td>66.1</td>
<td>50.8</td>
<td>43.8</td>
<td>57.7</td>
<td>71.3</td>
<td>57.0</td>
<td>43.3</td>
</tr>
<tr>
<td>CNNs</td>
<td>mean</td>
<td>58.5</td>
<td>71.1</td>
<td>58.4</td>
<td>46.4</td>
<td>57.6</td>
<td>71.4</td>
<td>57.6</td>
<td>43.2</td>
</tr>
<tr>
<td>LSTMs</td>
<td>mean</td>
<td>58.4</td>
<td>72.3</td>
<td>60.0</td>
<td>46.4</td>
<td>56.8</td>
<td>70.1</td>
<td>55.8</td>
<td>43.2</td>
</tr>
<tr>
<td>GRUs</td>
<td>mean</td>
<td>59.1</td>
<td>74.0</td>
<td>62.6</td>
<td>47.3</td>
<td>57.1</td>
<td>71.4</td>
<td>57.3</td>
<td>43.6</td>
</tr>
<tr>
<td>RCNNs</td>
<td>last</td>
<td>59.9</td>
<td>74.2</td>
<td>63.2</td>
<td>48.0</td>
<td>60.7</td>
<td>72.9</td>
<td>59.1</td>
<td>45.0</td>
</tr>
<tr>
<td>LSTMs + pre-train</td>
<td>mean</td>
<td>58.3</td>
<td>71.5</td>
<td>59.3</td>
<td>47.4</td>
<td>55.5</td>
<td>67.0</td>
<td>51.1</td>
<td>43.4</td>
</tr>
<tr>
<td>GRUs + pre-train</td>
<td>last</td>
<td>59.3</td>
<td>72.2</td>
<td>59.8</td>
<td>48.3</td>
<td>59.3</td>
<td>71.3</td>
<td>57.2</td>
<td>44.3</td>
</tr>
<tr>
<td>RCNNs + pre-train</td>
<td>last</td>
<td>61.3*</td>
<td>75.2</td>
<td>64.2</td>
<td>50.3*</td>
<td>62.3*</td>
<td>75.6*</td>
<td>62.0</td>
<td>47.1*</td>
</tr>
</tbody>
</table>

Further, our RCNN model also outperforms the other neural encoder models and the baselines of 62.0% on the test set, outperforming the word matching-based method BM25 by over 8 percent across all metrics. The ability of the RCNN model to outperform the other models indicates that the use of non-consecutive filters and a varying decay factor for the baselines and neural network models. For neural network models, we show the best average performance across 5 independent runs. For a fair comparison, we also pre-train 5 independent models for each configuration and then fine tune these models for each configuration, we report average performance across 5 independent runs.
# Classification Result

<table>
<thead>
<tr>
<th>Model</th>
<th>Fine</th>
<th>Binary</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Kalchbrener et al. 2014)</td>
<td>48.5</td>
<td>86.9</td>
</tr>
<tr>
<td>(Kim 2014)</td>
<td>47.4</td>
<td>88.1</td>
</tr>
<tr>
<td>(Tai et al. 2015)</td>
<td>51.0</td>
<td>88.0</td>
</tr>
<tr>
<td>(Kumar et al. 2016)</td>
<td>52.1</td>
<td>88.6</td>
</tr>
<tr>
<td>Constant, scalar decay</td>
<td>52.7</td>
<td>88.6</td>
</tr>
<tr>
<td>Gated decay</td>
<td>52.9</td>
<td>89.2</td>
</tr>
</tbody>
</table>

**Table 1:** Results on Stanford Sentiment Treebank.
Analysis

Does it help to model non-consecutive patterns?

![Chart showing the performance of different decay values on Dev and Test sets. The chart compares the accuracy of models with decay values 0.0, 0.3, and 0.5. The x-axis represents the Dev set accuracies ranging from 44.5% to 50.0%, while the y-axis represents the Test set accuracies ranging from 44.5% to 51.0%. The chart shows a scatter plot with points indicating the performance of each decay value.]