Learning to Label Stack Exchange Questions

Rafael Leano
Jose Picado
Zahra Iman
Problem

• Q/A sites get thousands of questions daily
• Questions must be labeled for easy retrieval
Task

- Automatically assign labels to questions using deep learning and natural language processing

How to efficiently iterate over each Entry in a Map?

If I have an object implementing the `Map` interface in Java and I wish to iterate over every pair contained within it, what is the most efficient way of going through the map?

1282

Will the ordering of elements depend on the specific map implementation that I have for the interface?
Task

- Automatically assign labels to questions using deep learning and natural language processing.

How to efficiently iterate over each Entry in a Map?

If I have an object implementing the `Map` interface in Java and I wish to iterate over every pair contained within it, what is the most efficient way of going through the map?

Will the ordering of elements depend on the specific map implementation that I have for the interface?
Approach

1. Convert title and questions to word vectors
   – GloVe: Global Vectors for Word Representation*
   – Pre-trained word vectors on Wikipedia 2014 + Gigaword 5
   – Dimension of vectors: 50

2. Train a long-short term memory (LSTM) model

3. Predict multiple labels for each example

*http://nlp.stanford.edu/projects/glove/
Architecture

word embeddings

Fully connected

Softmax

LSTM

LSTM

LSTM

LSTM

LSTM

\( w_1 \)

\( w_2 \)

\( w_n \)
Data

• Stack Exchange data* (~7GB):
  – Id, title, body, tags

• Statistics:
  – Questions: 6,034,195
  – Unique tags: 42,048
  – Tags/question: 1-5
  – Average no. of tags/question: 2.89

*https://www.kaggle.com/c/facebook-recruiting-iii-keyword-extraction/data
Reduced data

• Reduce number of tags:
  – Keep top 10 tags
  – Keep top 100 co-occurring tags with each top tag
  – Use only questions containing these tags
  – Remove stopwords, numbers and punctuation

• Statistics:
  – Questions: 1,220,004
  – Unique tags: 573
  – Maximum question length: 2694
First approach

• Input data:
  – $1,220,004 \times 2694 \times 50 = 164,334,538,800$

• Process in batches, reduce question size:
  – Batch size: 10,000 questions
  – Maximum question length: 512

• No improvement after each batch
Even more reduced data

- Reduce number of tags:
  - Keep top 10 tags
  - Keep top 10 co-occurring tags with each top tag
  - Use only questions containing these tags

- Statistics:
  - Questions: 582,331
  - Unique tags: 72
  - Maximum question length: 2694
Data used in experiments

- Questions: 200,000
- Unique tags: 72
- Maximum question length: 400

<table>
<thead>
<tr>
<th>Tag</th>
<th>Tag</th>
<th>Tag</th>
<th>Tag</th>
<th>Tag</th>
</tr>
</thead>
<tbody>
<tr>
<td>jquery</td>
<td>json</td>
<td>asp.net-mvc</td>
<td>eclipse</td>
<td></td>
</tr>
<tr>
<td>javascript</td>
<td>jquery-ajax</td>
<td>xml</td>
<td>spring</td>
<td></td>
</tr>
<tr>
<td>html</td>
<td>c#</td>
<td>wcf</td>
<td>hibernate</td>
<td></td>
</tr>
<tr>
<td>ajaxcss</td>
<td>.net</td>
<td>python</td>
<td>multithreading</td>
<td></td>
</tr>
<tr>
<td>php</td>
<td>winforms</td>
<td>java</td>
<td></td>
<td></td>
</tr>
<tr>
<td>jquery-ui</td>
<td>wpf</td>
<td>android</td>
<td>java-ee</td>
<td></td>
</tr>
<tr>
<td>asp.net</td>
<td>linq</td>
<td>swing</td>
<td></td>
<td>...</td>
</tr>
</tbody>
</table>
Baseline: N-grams + Logistic Regression

- Extract n-grams from title and body (cleaned text)
- Train a one-vs-rest classifier using Logistic Regression
- Implemented using scikit-learn (Python)

<table>
<thead>
<tr>
<th>N</th>
<th>Number of features</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-gram</td>
<td>1,171,964</td>
</tr>
<tr>
<td>{1,2}-gram</td>
<td>5,551,226</td>
</tr>
</tbody>
</table>
Results

- Training set: 180,000 questions (90%)
- Validation set: 20,000 questions (10%)
- Deep models:
  - 128 batch size
  - 18 epochs
  - Rmsprop
- Subset accuracy:
  - the set of predicted labels must *exactly* match the true set of labels

<table>
<thead>
<tr>
<th>Method</th>
<th>Validation Accuracy</th>
<th>Categorical Loss Entropy</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-gram + Logistic Regression</td>
<td>0.487</td>
<td>16.66</td>
</tr>
<tr>
<td>{1,2}-gram + Logistic Regression</td>
<td>0.518</td>
<td>16.09</td>
</tr>
<tr>
<td>2 layer LSTM</td>
<td>0.642</td>
<td>3.32</td>
</tr>
<tr>
<td>3 layer LSTM</td>
<td>0.641</td>
<td>3.48</td>
</tr>
</tbody>
</table>
Results (accuracy)

Questions: 100,000
3-layer LSTM

Questions: 200,000
2-layer LSTM
Results (categorical loss)

Questions: 100,000  
3-layer LSTM

Questions: 200,000  
2-layer LSTM
Results II (accuracy)

Questions: 200,000
3-layer LSTM

Questions: 200,000
2-layer LSTM
Results II (categorical loss)

Questions: 200,000
3-layer LSTM

Questions: 200,000
2-layer LSTM
References


• LSTM Networks for Sentiment Analysis. deeplearning.net.