An ensemble model for the machine reading comprehension dataset SQuAD

Summer report
Overview

- Problem definition
- Exploratory analysis
- Pipeline description
- Sentence ranking
- Answer extraction
- Implementation
Problem definition
Problem definition

- Implement a system capable of performing reading comprehension over SQuAD’s data set that outperforms the current state of the art.

- **SQuAD’s challenge:**
  - No candidate answers provided
  - A correct answer to a question can be any sequence of tokens from the given text
  - Q&A in SQuAD were created by humans, hence more realistic
Exploratory analysis
Exploratory analysis

- General statistics
- Lexical analysis
- Syntactic analysis
Complete dataset

- 536 Wikipedia articles
- 108K QA pairs
- Training, dev and test
- Hierarchical view:

```
<table>
<thead>
<tr>
<th>Article n</th>
</tr>
</thead>
<tbody>
<tr>
<td>Passage_{m,n}</td>
</tr>
<tr>
<td>Question_{1,m,n}</td>
</tr>
<tr>
<td>Question_{2,m,n}</td>
</tr>
<tr>
<td>Question_{3,m,n}</td>
</tr>
<tr>
<td>Question_{4,m,n}</td>
</tr>
<tr>
<td>Question_{5,m,n}</td>
</tr>
<tr>
<td>Answer_{1,m,n}</td>
</tr>
<tr>
<td>Answer_{2,m,n}</td>
</tr>
<tr>
<td>Answer_{3,m,n}</td>
</tr>
<tr>
<td>Answer_{4,m,n}</td>
</tr>
<tr>
<td>Answer_{5,m,n}</td>
</tr>
</tbody>
</table>
```

- \# words
  - Passages: \~88K (98\% without stop words)
  - Questions: \~1K (93\% w/o stop words)
  - Answers: \~0.5K (93\% w/o stop words)

Training dataset

- 378 Wikipedia articles
- \~42 passages per article
- 5 questions per passage
- 1 answer per question
- \~80K QA pairs

Model evaluation

- Output: sequence of tokens
- Measures: Exact match, F1

Vocabulary Size
>99% of the questions are factoid; >50% are what questions
Questions length is similar; answers to *why* and *other* questions show length variation.
Questions are larger than answers; *why* questions have the largest answers but represent <5%
Exploratory analysis

- General statistics
- Lexical analysis
- Syntactic analysis
There exists a lexical similarity 0.3-0.4 between passages of the same article

* Random passages were extracted from all the articles

** Measured as cosine similarity
This similarity is independent of the length of the passage
LDA analysis varying number of words and topics showed the following persistent topics

- history
- government
- nation-state
- sports
- art
Exploratory analysis

- General statistics
- Lexical analysis
- Syntactic analysis
  - Embeddings
    - Word
    - Sentence
    - Paragraph
Word embeddings pipeline

- Word embeddings
- Dimensionality reduction
- Clustering
- Visualization

- Linear
- Syntactic
- Topical
- t-SNE
- k-means
- scatterplot
Models

- Glove
- Skip-gram

Parameters

- Window size
- Vector size
GLOVE Linear Embedding

- Min words in voc = 100
- Size of vectors = 100, 300, 500
- Size of window = 5, 15, 20
GLOVE Linear Embedding

Window Size = 15
Vector Size = 100
GLOVE Linear Embedding

Window Size = 15
Vector Size = 100
Cluster = 2
GLOVE Linear Embedding

Window Size = 15
Vector Size = 100
Cluster = 4
GLOVE Linear Embedding

Window Size = 15
Vector Size = 100
Cluster = 5
GLOVE Linear Embedding

Window Size = 20
Vector Size = 100
GLOVE Linear Embedding

Window Size = 20
Vector Size = 100
Cluster = 5
GLOVE Linear Embedding

Window Size = 20
Vector Size = 100
Cluster = 6
GLOVE Linear Embedding

Window Size = 5
Vector Size = 100
GLOVE Linear Embedding

Window Size = 5
Vector Size = 100
Cluster = 5
Window Size = 5
Vector Size = 100
Cluster = 6
GLOVE Linear Embedding

Window Size = 15
Vector Size = 500
GLOVE Linear Embedding

Window Size = 15
Vector Size = 500
Cluster = 1
GLOVE Linear Embedding

Window Size = 15
Vector Size = 500
Cluster = 2
GLOVE  Syntactic Embedding

- Min words in voc = 100
- Size of vectors = 100, 300, 500
- Size of window = 5, 15, 20
GLOVE Syntactic Embedding

Window Size = 15
Vector Size = 300
GLOVE Syntactic Embedding

Window Size = 15
Vector Size = 300
Cluster = 1

Captures different relations
GLOVE Syntactic Embedding

Window Size = 15
Vector Size = 300
Cluster = 2

Captures different relations
GLOVE Topic Embedding

**Topic 1st:**
- jewish 0.022814
- jews 0.021276
- communities 0.009680
- see 0.005708
- judaism 0.005644
- orthodox 0.005516
- community 0.005324
- hebrew 0.005068
- israel 0.003658
- palestine 0.001864
- synagogue 0.001544
- persecution 0.001416
- jerusalem 0.001352
- group 0.001224
- holocaust 0.001224
- judah 0.001160

**Topic 5th:**
- pope 0.014170
- paul 0.008777
- john 0.006652
- cardinal 0.006597
- cardinals 0.005726
- bishops 0.005508
- athanasius 0.005344
- vi 0.005072
- rome 0.004963
- bishop 0.004309
- pius 0.003819
- see 0.003547
- vatican 0.003492
- papal 0.003056
- order 0.003002
- saint 0.002675

**Topic 9th:**
- economic 0.013044
- financial 0.009602
- economy 0.008634
- government 0.008365
- development 0.008311
- industry 0.007559
- public 0.007317
- world 0.005945
- trade 0.005918
- also 0.005649
- international 0.005596
- countries 0.005569
- production 0.005112
- sector 0.004762
- crisis 0.004762
- organization 0.004708
- Min words in voc = 100
- Size of vectors = 300
- Size of window = 15
GLOVE Topic Embedding

Window Size = 15
Vector Size = 300
GLOVE Topic Embedding

Window Size = 15
Vector Size = 300
Cluster = 1

GLOVE with topic capture

Broad topics
GLOVE Topic Embedding

Window Size = 15
Vector Size = 300
Cluster = 6

GLOVE with topic capture
Broad topics
Exploratory analysis

- General statistics
- Lexical analysis
- Syntactic analysis
  - Embeddings
    - Word
    - Sentence
    - Paragraph
Similar words to 6W+how questions

**what** (80% of questions)
1. which: 0.67
2. where
3. represent
4. resemble
5. supports
6. origins
7. institution
8. protect
9. formal
10. mainly

**who**
1. succeeded: 0.78
2. successor
3. supports
4. prevented
5. group
6. party
7. freemasons
8. criticized
9. rebel
10. toward

Python’s Doc2Vec on questions
- Min_count=10
- Window Size = 10
- Vector Size = 100
### Similar words to 6W+how questions

<table>
<thead>
<tr>
<th>how</th>
<th>which</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. there: 0.73</td>
<td>1. named: 0.67</td>
</tr>
<tr>
<td>2. about</td>
<td>2. dominated</td>
</tr>
<tr>
<td>3. people</td>
<td>3. consisted</td>
</tr>
<tr>
<td>4. lines</td>
<td>4. formed</td>
</tr>
<tr>
<td>5. live</td>
<td>5. mayor</td>
</tr>
<tr>
<td>6. days</td>
<td>6. divides</td>
</tr>
<tr>
<td>7. million</td>
<td>7. Somali</td>
</tr>
<tr>
<td>8. many</td>
<td>8. dominant</td>
</tr>
<tr>
<td>9. millions</td>
<td>9. formerly</td>
</tr>
<tr>
<td>10. killed</td>
<td>10. reform</td>
</tr>
</tbody>
</table>
Similar words to 6W+how questions

when
1. why: 0.71
2. son
3. John
4. succeeded
5. leave
6. revolution
7. richard
8. constantinople
9. ask
10. before

where
1. v
Similar words to 6W+how questions

why
1. stepper: 0.82
2. absorb
3. doing
4. mark
5. without
6. efficacy
7. genes
8. can
9. insects
10. maintain
Exploratory analysis

- General statistics
- Lexical analysis
- Syntactic analysis
  - Embeddings
    - Word
    - Sentence
    - Paragraph
Paragraph embeddings detect similarities between words

Synonym identification:

- \( \text{sim}([\text{'college'}, \text{'professor'}], [\text{'university'}, \text{'teacher'}]) = 0.92 \)
- \( \text{sim}([\text{'marriage'}, \text{'husband'}, \text{'baby'}], [\text{'wife'}, \text{'wedding'}, \text{'children'}]) = 0.85 \)
- \( \text{sim}([\text{'house'}, \text{'residence'}, \text{'bed'}, \text{'accommodation'}, \text{'address'}], [\text{'shelter'}, \text{'mansion'}, \text{'home'}, \text{'place'}]) = 0.77 \)

Python's Doc2Vec on paragraphs

Min_count=10
Window Size = 10
Vector Size = 100
This analysis also detects non-related terms and analogies

Non-related terms identification:

- similarity('husband', 'floor') = 0.30
- similarity('night', 'chicken') = 0.29
- similarity('computer', 'city') = 0.22

Analogies

- woman is to king as man is to...? prince
- Most similar to “queen”: Madonna, widow, performed
- Most similar to “man”: said, wrote, god

Python’s Doc2Vec on paragraphs

Min_count=10
Window Size = 10
Vector Size = 100
The topics found with LDA can be refined using paragraph embeddings

**LDA:**

- church + roman + first + emperor ~ history
- state + govern + force + war ~ government

Most similar words to LDA keywords:

**Roman Empire?**
1. rome: 0.86
2. byzantine
3. centuries
4. patriarch
5. 14th
6. survived
7. 12th
8. successors
9. constantine
10. succession

**war?**
1. government: 0.85
2. administration
3. sovereign
4. military
5. suppress
6. forces
7. initiated
8. supported
9. organized
10. urged
The topics found with LDA can be refined using paragraph embeddings

**LDA:**

- city + new + state + area + unit ~ nation-state
- game + team + play ~ sports

Most similar words to LDA keywords:

1. located: 0.86
2. metropolitan
3. headquarters
4. county
5. designated
6. operated
7. downtown
8. currently
9. main
10. serves

1. championship: 0.89
2. games
3. players
4. fans
5. exhibition
6. afl
7. matches
8. teams
9. nfl
10. super
The topics found with LDA can be refined using paragraph embeddings

*LDA:*

music + film + record ~ art

Most similar words to LDA keywords:

1. films: 0.9
2. featured
3. movie
4. studio
5. singers
6. guitar
7. songs
8. artist
9. albums
10. hip-hop
Pipeline description
High level baseline pipeline

Sentence ranking ➔ Answer extraction ➔ Learning ➔ Evaluation
Sentence ranking
High level baseline pipeline
Sentence Ranking

The whole idea of sentence ranking is to exploit lexical and syntactical similarities between the question and the answer passage to obtain the sentence with the highest likelihood of being the answer.
Sentence Ranking Convolutional Neural Networks

Convolutional neural network model for reranking pairs of short texts:

- Learn optimal vector representation of Q-D
- Learn a similarity function between Q-D vectors
Sentence Ranking Convolutional Neural Networks

Figure 2: Our deep learning architecture for reranking short text pairs.
Sentence Ranking Convolutional Neural Networks

Sentences are represented as sequences of words, where each word is an $|s|$ dimensional continuous representation.

$$S = \begin{bmatrix} w_1 & \cdots & w_{|s|} \end{bmatrix}$$

A filter $f$ is applied to the sequence in order to capture interactions among words.

$$c_i = (s * f)_i = s_{[i-m+1:i]}^T \cdot f = \sum_{k=i}^{i+m-1} s_k f_k$$
After this is done, a nonlinear activation function, ReLU in this case, is applied to every $c_i$ and the results are pooled together via max pooling into a single $c_{\text{pooled}}$ array representation.

$$c_{\text{pooled}} = \begin{bmatrix} \text{pool}(\alpha(c_1 + b_1 \ast e)) \\ \ldots \\ \text{pool}(\alpha(c_n + b_n \ast e)) \end{bmatrix}$$
Once these representations are obtained for each sentence $x_d$ and each query $x_q$, a $x_{\text{sim}}$ score is obtained by $x_d \cdot M x_q$ and an $x_{\text{join}}$ is created by simple concatenation. Each $x_{\text{join}}$ is passed through a hidden layer to exploit interactions among its different components, and finally a softmax is used for the ranking.
The model is trained to minimize the cross-entropy function:

\[
C = -\log \prod_{i=1}^{N} p(y_i | q_i, d_i) + \lambda \| \theta \|_2^2 = -\sum_{i=1}^{N} [y_i \log a_i + (1 - y_i) \log (1 - a_i)] + \lambda \| \theta \|_2^2,
\]

where \( a \) is the output of the softmax and \( \theta \) contains all the parameters of the network:

\[
\theta = \{ W; F_q; b_q; F_d; b_d; M; w_h; b_h; w_s; b_s \}
\]

Regularization is used to avoid overfitting and stochastic gradient descent to cope with the non convexity of the problem.
In our model, we added an hybrid vector representation that used both, the representation trained over the AQUAINT corpus (to obtain the most general context of each word), and over the SQUAD dataset (to obtain the particular uses of each word). We also used Jaccard similarity as a proxy of relevance judgments, and we added topic information to the $x_{\text{joint}}$ representation.
Sentence Ranking  BM25 & Jaccard similarity

Another approach that uses only lexical similarity, under the bag of words was applied, namely BM25 and Jaccard similarity:

- **BM25**

  \[
  \text{score}(D, Q) = \sum_{i=1}^{n} \text{IDF}(q_i) \cdot \frac{f(q_i, D) \cdot (k_1 + 1)}{f(q_i, D) + k_1 \cdot \left(1 - b + b \cdot \frac{|D|}{\text{avgdl}} \right)}
  \]

  \[
  \text{IDF}(q_i) = \log \frac{N - n(q_i) + 0.5}{n(q_i) + 0.5}
  \]

- **Jaccard similarity**

  \[
  J(A, B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}
  \]
Sentence Ranking Results

<table>
<thead>
<tr>
<th>MODEL</th>
<th>MRR Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Convolutional Neural Networks</td>
<td>.25</td>
</tr>
<tr>
<td>BM25</td>
<td>.71</td>
</tr>
<tr>
<td>Jaccard</td>
<td>.76</td>
</tr>
</tbody>
</table>

We believe that the bad results of the ConvNets is due to underfitting of the training data.
High level baseline pipeline

Sentence ranking  Answer extraction  Learning  Evaluation
Answer extraction
Idea: Use *features* that extract lexical, syntactical and semantical structure of sentence, question and answer to train a classifier.

For each word in candidate answer sentence:

- indicator of right neighbor in question
- right neighbor NER
- right neighbor POS
- word Animacy
- word Gender
- word NER
- word Number
- word POS
- word type
- dependency with father
- indicator father in question
- father NER
- father POS
- indicator of word in question
- indicator of left neighbor in question
- left neighbor NER
- left neighbor POS
- question type
Answer extraction  Random Forests

Example: It

(False, u'\textsc{It}', u'PRP', u'O', False, 'whom', '', '', '', u'is', u'VBZ', u'O', False, False, u'replica', u'NN', u'O', u'nsubj', False, False, u'INANIMATE', u'SINGULAR', u'NEUTRAL', u'PRONOMINAL')
Answer extraction Random Forests

Model
- 1000 trees
- 5 variables per cut
- Gini criterion

Results
Training:
F1 score = .49
precision = .62

Test:
F1 score = .47
precision = .6
Random Forests
Pipeline implementation
Pipeline implementation

Our pipeline implementation supports:

- An end to end pipelined execution.
- Model training
- Model testing
- Interactive Mode
Pipeline implementation

Model training:

The system allows you to choose the number of sentences to be considered as part of the answer as well as the number of instances used on the training phase.
Model testing:

It also gives you the option to train or test the model. And provides a final evaluation with Stanford's script.
Pipeline implementation

Model interactive mode:

Finally, to enable testing of new models, the system also supports interactive mode.
Pipeline implementation

Model interactive mode:

Finally, to enable testing of new models, the system also supports interactive mode.
Pipeline implementation

End to end execution results evaluated under Stanford's metric:

{"f1": 0.20368373764600187, "exact_match": 0.07547169811320754}

<table>
<thead>
<tr>
<th></th>
<th>Exact Match</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Dev</td>
<td>Test</td>
</tr>
<tr>
<td>Random Guess</td>
<td>1.1%</td>
<td>1.3%</td>
</tr>
<tr>
<td>Sliding Window</td>
<td>13.2%</td>
<td>12.5%</td>
</tr>
<tr>
<td>Sliding Win. + Dist.</td>
<td>13.3%</td>
<td>13.0%</td>
</tr>
<tr>
<td>Logistic Regression</td>
<td>40.0%</td>
<td>40.4%</td>
</tr>
<tr>
<td>Human</td>
<td>80.3%</td>
<td>77.0%</td>
</tr>
</tbody>
</table>
References

- Tomas Mikolov, Quoc V. Le and Ilya Sutskever. Exploiting Similarities among Languages for Machine Translation.
References

References

