Distributed Representations of Sentences and Paragraphs

Presented by: Gaddipati, Asish Kumar
Authors: Quoc Le and Tomas Mikolov
NLP Tasks

- Data representation plays a major role for NLP tasks like
  - Sentiment Analysis
    - Ex: Classifying movie reviews as positive and negative.
  - Text Classification
    - Ex: Classifying documents into multiple categories.
  - Information retrieval
    - Ex: Retrieving relevant results for given search query.
How to represent the input?
- Document: Document Classification Task
- Movie review: Sentiment Analysis Task

**Common Approaches**
- Bag of words
- Bag of n-grams
Problems

The problems of representing the input with these approaches are:

- The vectors are sparse and highly dimensional.
- Unable to capture the semantics.
  - Similarity (Powerful, Strong) >> Similarity (Powerful, Paris)
- Does not capture the order of the words.
  - Ex: Did you ever see a cake walk?
    Did you ever see a walk on cake?
Doc2Vec

- Generalization of word2vec to variable length sized text like Documents, Paragraphs, Sentences etc.
  (Word2vec package has skip gram and continuous bag of words model)
- Fixed length representation of the input.
- Two Versions
  - Paragraph Model – Distributed memory
    - Similar to continuous bags of words model
  - Paragraph Model – Distributed bag of words
    - Similar to skip gram model
Paragraph Model – Distributed Memory

Given a sequence of paragraphs \( s_1, s_2, \ldots, s_q \) and training words \( w_1, w_2, w_3, \ldots, w_T \), maximize

\[
\frac{1}{T} \sum_{t=1}^{T} \sum_{-c \leq k \leq c, k \neq 0} \log(p(w_t | s_j, w_{t-k}, w_{t+k}))
\]

where prediction task is done by softmax regression

\[
p(w_t | s_j, w_{t-k}, w_{t+k}) = \frac{e^{y_{w_t}}}{\sum_i e^{y_i}}
\]

Each of \( y_i \) is un-normalized log-probability for each output word \( i \), computed as

\[
y = U*h(s_j, w_{t-k}, w_{t+k}; W, D) + b
\]

- \( U, b \) are Soft max parameters
- \( h \) is constructed by \textit{concatenation} or average of word vectors.
- Each column of \( W \) is word embedding of a Word.
- Each column of \( D \) is paragraph embedding of a Paragraph.
Paragraph Model – Distributed Memory

Training

- Data: N paragraphs and M Words
- Randomly Initialize D paragraph matrix and W word embedding’s matrix
- For each paragraph
  - Choose a context window of size c
  - Predict the next word using paragraph vector and contextual windows
  - Slide the context window over the paragraph by keeping the paragraph vector unchanged.
  - Update the parameters using stochastic gradient descent and back propagation

After Training: We will learn Paragraph Matrix $D_{N \times P}$ and Word embedding’s Matrix $W_{Q \times M}$ and classifier parameters $U$ and $b$.

At Prediction: Given new Paragraphs, Add new columns to D and update D using stochastic gradient descent and previously learnt W, U and b.
Paragraph Model – Distributed Memory

- Similar to continuous bag of words model
- For example consider 2 paragraphs and window size of 2
  - P1: The cat sat on the mat
  - P2: I ate potato crisps for evening snack
Paragraph Model - Distributed bag of words

- Force the model to predict the words randomly sampled from paragraph as output by only using paragraph vectors
- Simpler and more memory efficient
- This is similar to skip-gram model
Experiments

The paragraph vectors are tested for
- Sentiment Analysis task on
  - Stanford data set
    - Each sample has one sentence
  - IMDB Data set
    - Each sample has more than one sentence

<table>
<thead>
<tr>
<th>Data Set</th>
<th>#Train Samples</th>
<th>#Validation</th>
<th>#Test samples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stanford dataset</td>
<td>8544</td>
<td>1101</td>
<td>2210</td>
</tr>
<tr>
<td>IMDB data set</td>
<td>25000(labelled)</td>
<td>50000(un labelled)</td>
<td>25000(labelled)</td>
</tr>
</tbody>
</table>

- Information Retrieval task on
  - Data set of Triplets, of search engine resulted snippets for 1,000,000 popular queries.
  - Each triplet will have 2 snippets of a given query and 3rd one is randomly sampled from results
Stanford Sentiment Analysis Dataset

Evaluated on coarse as well as on fine grained labels
- Considered each sub phrase also as sentence
- Fixed window size as 8 upon cross validation
- Learn 400-dim vector representing sentences using DM and DBOW
- Append to get 800-dim vector for classification task
- Predict test set paragraph vectors from frozen train set word vectors.
- Use a classifier to predict the sentiment
Stanford Dataset: Results

- As expected, the bag of words/n-grams (NB, SVM, BiNB) gave poor results as they don't capture syntactic and semantics features.
- Averaging the word vectors didn’t improve the performance, it's in fact decreased.
- More complex models (RNN’s, RNTN) need preprocessing of data like parsing and they work on single sentences only.
- Paragraph vector outperforms with 16% relative improvement and does not need any preprocessing like above.

<table>
<thead>
<tr>
<th>Model</th>
<th>Error rate (Positive/Negative)</th>
<th>Error rate (Fine-grained)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Naïve Bayes (Socher et al., 2013b)</td>
<td>18.2%</td>
<td>59.0%</td>
</tr>
<tr>
<td>SVMs (Socher et al., 2013b)</td>
<td>20.6%</td>
<td>59.3%</td>
</tr>
<tr>
<td>Bigram Naïve Bayes (Socher et al., 2013b)</td>
<td>16.9%</td>
<td>58.1%</td>
</tr>
<tr>
<td>Word Vector Averaging (Socher et al., 2013b)</td>
<td>19.9%</td>
<td>67.3%</td>
</tr>
<tr>
<td>Recursive Neural Network (Socher et al., 2013b)</td>
<td>17.6%</td>
<td>56.8%</td>
</tr>
<tr>
<td>Matrix Vector-RNN (Socher et al., 2013b)</td>
<td>17.1%</td>
<td>55.6%</td>
</tr>
<tr>
<td>Recursive Neural Tensor Network (Socher et al., 2013b)</td>
<td>14.6%</td>
<td>54.3%</td>
</tr>
<tr>
<td><strong>Paragraph Vector</strong></td>
<td><strong>12.2%</strong></td>
<td><strong>51.3%</strong></td>
</tr>
</tbody>
</table>
IMDB Movie review data set

- Train DM and DBOW and then append to get 400-dim paragraph vectors for 75,000 training samples
- Fixed the window size as 10 upon cross validation
- Predict test set paragraph vectors from frozen train set word vectors
- Use a classifier to predict the sentiment

Results

- For long sentences like paragraphs Bags of words did reasonably well.
- Paragraph vectors outperforms all the state of the art (NBSVM-bi) with a relative improvement of 15%.

<table>
<thead>
<tr>
<th>Model</th>
<th>Error rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>BoW (bnc) (Maas et al., 2011)</td>
<td>12.20%</td>
</tr>
<tr>
<td>BoW (bΔt’c) (Maas et al., 2011)</td>
<td>11.77%</td>
</tr>
<tr>
<td>LDA (Maas et al., 2011)</td>
<td>32.58%</td>
</tr>
<tr>
<td>Full+BoW (Maas et al., 2011)</td>
<td>11.67%</td>
</tr>
<tr>
<td>Full+Unlabeled+BoW (Maas et al., 2011)</td>
<td>11.11%</td>
</tr>
<tr>
<td>WRRBM (Dahl et al., 2012)</td>
<td>12.58%</td>
</tr>
<tr>
<td>WRRBM + BoW (bnc) (Dahl et al., 2012)</td>
<td>10.77%</td>
</tr>
<tr>
<td>MNB-uni (Wang &amp; Manning, 2012)</td>
<td>16.45%</td>
</tr>
<tr>
<td>MNB-bi (Wang &amp; Manning, 2012)</td>
<td>13.41%</td>
</tr>
<tr>
<td>SVM-uni (Wang &amp; Manning, 2012)</td>
<td>13.05%</td>
</tr>
<tr>
<td>SVM-bi (Wang &amp; Manning, 2012)</td>
<td>10.84%</td>
</tr>
<tr>
<td>NBSVM-uni (Wang &amp; Manning, 2012)</td>
<td>11.71%</td>
</tr>
<tr>
<td>NBSVM-bi (Wang &amp; Manning, 2012)</td>
<td>8.78%</td>
</tr>
<tr>
<td><strong>Paragraph Vector</strong></td>
<td><strong>7.42%</strong></td>
</tr>
</tbody>
</table>
Information Retrieval

- **Goal**: To identify if a document should be retrieved given a search query
- **For Example**:

  **Snippet 1**: calls from ( 000 ) 000 - 0000 . 3913 calls reported from this number, according to 4 re-ports the identity of this caller is American airlines.

  **Snippet 2**: do you want to find out who called you from +1 000 - 000 - 0000 , +1 0000000000 or ( 000 ) 000 - 0000 ? see reports and share information you have about this caller.

  **Snippet 3**: allina health clinic patients for your convenience, you can pay your allina health clinic bill online, pay your clinic bill now, question and answers...

- Identify the triplets that are results of a given query?
  
  Similarity(Snippet1, Snippet2) >> Similarity(Snippet1/2, Snippet3)

- Paragraph vector outperformed all the previous methods

<table>
<thead>
<tr>
<th>Model</th>
<th>Error rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vector Averaging</td>
<td>10.25 %</td>
</tr>
<tr>
<td>Bag-of-words</td>
<td>8.10 %</td>
</tr>
<tr>
<td>Bag-of-bigrams</td>
<td>7.28 %</td>
</tr>
<tr>
<td>Weighted Bag-of-bigrams</td>
<td>5.67 %</td>
</tr>
<tr>
<td>Paragraph Vector</td>
<td><strong>3.82 %</strong></td>
</tr>
</tbody>
</table>
Observations

- DM is consistently better than DBOW and combination of both is recommended as it often work consistently better.
- While combining, concatenation is preferred over sum as it preserves ordering information.
- Paragraph vector is expensive but it can be done parallel during test time.