Distributed Representations of Words and Phrases and their Compositionality

2013b

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, Jeffrey Dean

Seminar “Selected Topics in Semantics and Discourse”, presenter Yauhen Klimovich, tutor Prof. Manfred Pinkal
What is the distributed representations of words?
What is the distributed representations of words?

a vector

e.g. 300-d vector

\((3.4, 0.45, \ldots, 7.4, 5.63)\)

\((x_1, x_2, \ldots, x_{299}, x_{300})\)
What is the distributed representations of words?

a vector

word embeddings, word projections

e.g. 300-d vector

\((3.4, 0.45, \ldots, 7.4, 5.63)\)

\((x_1, x_2, \ldots, x_{299}, x_{300})\)
What is the distributed representations of words?

a vector

word embeddings, word projections

e.g. 300-d vector

\((3.4, 0.45, \ldots, 7.4, 5.63)\)

\((x_1, x_2, \ldots, x_{299}, x_{300})\)

Good for similarity measure
And phrases?
And phrases?

Meaningful word compounds
And phrases?

Meaningful word compounds

e.g. ‘Tesla Motors’
   is neither Tesla nor Motors,
   ‘Toronto Maple Leafs’
   is not Toronto and a maple and leafs
And phrases?

Meaningful word compounds

e.g. ‘Tesla Motors’
is neither Tesla nor Motors,
‘Toronto Maple Leafs’
is not Toronto and a maple and leafs

Good for accuracy
What is compositionality?
What is compositionality?

Math operations on word vectors
What is compositionality?

Math operations on word vectors

\[ \text{vec(“Germany”) + vec(“capital”) \approx vec(“Berlin”)} \]
What is compositionality?

Math operations on word vectors

\[ \text{vec("Germany") + vec("capital") } \approx \text{vec("Berlin")} \]
\[ \text{vec("Steve Ballmer") - vec("Microsoft") + vec("Google") } \approx \text{vec("Larry Page")} \]
What is compositionality?

Math operations on word vectors

\[
\text{vec(“Germany”)} + \text{vec(“capital”)} \approx \text{vec(“Berlin”)}
\]
\[
\text{vec(“Steve Ballmer”)} - \text{vec(“Microsoft”)} + \text{vec(“Google”)} \approx \text{vec(“Larry Page”)}
\]

Basic operations can give us better results
Agenda

• Skip-gram in details
• Improvements for skip-gram
• Phrases
• Evaluation
Why is this paper important?

**Improvements** for vector quality and training speed
Why is this paper important?

**Improvements** for vector quality and training speed

Continuous skip-gram

“Efficient Estimation of Word Representations in Vector Space”  
[Mikolov et al, 2013]
Why is this paper important?

**Improvements** for
vector quality and training speed

Continuous skip-gram

"Efficient Estimation of Word Representations in Vector Space"
[Mikolov et al, 2013]

No dense matrix multiplication!
Why is this paper important?

**Improvements** for
vector quality and training speed

Continuous skip-gram

“Efficient Estimation of Word Representations in Vector Space”
[Mikolov et al, 2013]

No dense matrix multiplication!

Efficient!
We talk about “Distributed Representations of Words and Phrases and their Compositionality” (Mikolov et al)
Why is this paper important?

**Improvements** for
vector quality and training speed

Continuous skip-gram

“Efficient Estimation of Word Representations in Vector Space”
[Mikolov et al, 2013]

No dense matrix multiplication!

Efficient!

What is new?

- Negative sampling approach
- Subsampling of frequent words
- Phrases (Tesla Motors, Silicon Valley)
Skip-gram model
Skip-gram model

We talk about “Distributed Representations of Words and Phrases and their Compositionality” (Mikolov et al)
Skip-gram model

We talk about “Distributed Representations of Words and Phrases and their Compositionality” (Mikolov et al)
We talk about “Distributed Representations of Words and Phrases and their Compositionality” (Mikolov et al)

Skip-gram model

inefficient, let’s approximate
We talk about “Distributed Representations of Words and Phrases and their Compositionality” (Mikolov et al).

Skip-gram model

inefficient, let’s approximate

Hierarchical softmax
binary Huffman tree
(short codes to frequent words)
Hierarchical softmax

Efficient way to compute softmax
Hierarchical softmax

Efficient way to compute softmax

\[
p(w|w_I) = \prod_{j=1}^{L(w)-1} \sigma \left( [n(w, j + 1) = ch(n(w, j))] \cdot v'_n(w, j)^\top v_{w_I} \right)
\]

\[
S(t) = \frac{1}{1 + e^{-t}}.
\]

for normalization

\[
[x] = \begin{cases} 
1 & \text{if } x \text{ is true;} \\
-1 & \text{otherwise.}
\end{cases}
\]

better for infrequent words, fast training
Hierarchical softmax

Efficient way to compute softmax

\[
p(w|w_I) = \prod_{j=1}^{L(w)-1} \sigma \left( [n(w, j + 1) = ch(n(w, j))] \cdot v'_n(w, j)^\top v_{w_I} \right)
\]

\[
S(t) = \frac{1}{1 + e^{-t}}.
\]

for normalization

Max logW for each word

better for infrequent words, fast training

We talk about “Distributed Representations of Words and Phrases and their Compositionality” (Mikolov et al)
Hierarchical softmax

Efficient way to compute softmax

\[
p(w | w_I) = \prod_{j=1}^{L(w)-1} \sigma \left( \left[ n(w, j+1) = ch(n(w, j)) \right] \cdot v'_{n(w, j)} ^T v_{w_I} \right)
\]

\[
S(t) = \frac{1}{1 + e^{-t}}.
\]

for normalization

\[
[x] = \begin{cases} 
1 & \text{if } x \text{ is true;} \\
-1 & \text{otherwise.}
\end{cases}
\]

Max \( \log W \) for each word

better for infrequent words, fast training

Structure of the tree is important
Negative sampling

idea is based on Noise Contrastive Estimation (NCE)

$$\log \sigma(v'_w o^T v_w l) + \sum_{i=1}^{k} \mathbb{E}_{w_i \sim P_n(w)} \left[ \log \sigma(-v'_w i^T v_w l) \right]$$
We talk about “Distributed Representations of Words and Phrases and their Compositionality” (Mikolov et al)

Negative sampling

idea is based on Noise Contrastive Estimation (NCE)

\[ \log \sigma(\mathbf{u}_w^T \mathbf{v}_w) + \sum_{i=1}^{k} \mathbb{E}_{w_i \sim P_n(w)} \left[ \log \sigma(-\mathbf{u}_w^T \mathbf{v}_w) \right] \]

noise distribution:
best result is given by \( U(w)^{(\frac{3}{4})} \)

trained classifier
Negative sampling

The idea is based on Noise Contrastive Estimation (NCE):

\[
\log \sigma(v'_{w_O}^T v_{w_I}) + \sum_{i=1}^{k} \mathbb{E}_{w_i \sim P_n(w)} \left[ \log \sigma(-v'_{w_i}^T v_{w_I}) \right]
\]

noise distribution:
best result is given by $U(w)^{(3/4)}$

trained classifier

better for frequent words,
better with low dimensional vectors
Subsampling of frequent words

\[ P(w_i) = 1 - \sqrt{\frac{t}{f(w_i)}} \]
Subsampling of frequent words

\[ P(w_i) = 1 - \sqrt{\frac{t}{f(w_i)}} \]

Frequency of \( w_i \)

threshold \( 1/10^5 \)

improves the accuracy of the learned vectors of the rare words
Subsampling of frequent words

\[ P(w_i) = 1 - \sqrt{\frac{t}{f(w_i)}} \]

threshold \(1/10^5\)

Frequency of \(w_i\)

improves the accuracy of the learned vectors of the rare words

cutting them off, context is larger
Evaluation

Analogical reasoning task:
- syntactic analogy
- semantic analogy

Data
- 1b words;
- cut out infrequent words(<5t),
- they got $|Voc| = 692K$

about 19.5k samples

- Berlin Germany Bern Switzerland
- boy girl brother sister
- amazing amazingly apparent apparently
- acceptable unacceptable certain uncertain
- cold colder great greater
- Europe euro Romania leu

- bright brightest sharp sharpest
- code coding jump jumping
- Belarus Belorussian Germany German
- flying flew enhancing enhanced
- car cars cat cats
- enhance enhances work works
Evaluation (results)

Analogical reasoning task:
- syntactic analogy
- semantic analogy

Data
- 1b words;
- cut out infrequent words (<5t);
- they got |Voc| = 692K

<table>
<thead>
<tr>
<th>Method</th>
<th>Time [min]</th>
<th>Syntactic [%]</th>
<th>Semantic [%]</th>
<th>Total accuracy [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>NEG-5</td>
<td>38</td>
<td>63</td>
<td>54</td>
<td>59</td>
</tr>
<tr>
<td>NEG-15</td>
<td>97</td>
<td>63</td>
<td>58</td>
<td>61</td>
</tr>
<tr>
<td>HS-Huffman</td>
<td>41</td>
<td>53</td>
<td>40</td>
<td>47</td>
</tr>
<tr>
<td>NCE-5</td>
<td>38</td>
<td>60</td>
<td>45</td>
<td>53</td>
</tr>
</tbody>
</table>

The following results use 10^-5 subsampling

<table>
<thead>
<tr>
<th>Method</th>
<th>Time [min]</th>
<th>Syntactic [%]</th>
<th>Semantic [%]</th>
<th>Total accuracy [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>NEG-5</td>
<td>14</td>
<td>61</td>
<td>58</td>
<td>60</td>
</tr>
<tr>
<td>NEG-15</td>
<td>36</td>
<td>61</td>
<td>61</td>
<td>61</td>
</tr>
<tr>
<td>HS-Huffman</td>
<td>21</td>
<td>52</td>
<td>59</td>
<td>55</td>
</tr>
</tbody>
</table>

Table 1: Accuracy of various Skip-gram 300-dimensional models on the analogical reasoning task as defined in [8]. NEG-k stands for Negative Sampling with k negative samples for each positive sample; NCE stands for Noise Contrastive Estimation and HS-Huffman stands for the Hierarchical Softmax with the frequency-based Huffman codes.
Evaluation(results)

Analogical reasoning task:
- syntactic analogy
- semantic analogy

Data
- 1b words;
- cut out infrequent words(<5t),
- they got |Voc| = 692K

<table>
<thead>
<tr>
<th>Method</th>
<th>Time [min]</th>
<th>Syntactic [%]</th>
<th>Semantic [%]</th>
<th>Total accuracy [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>NEG-5</td>
<td>38</td>
<td>63</td>
<td>54</td>
<td>59</td>
</tr>
<tr>
<td>NEG-15</td>
<td>97</td>
<td>63</td>
<td>58</td>
<td>61</td>
</tr>
<tr>
<td>HS-Huffman</td>
<td>41</td>
<td>53</td>
<td>40</td>
<td>47</td>
</tr>
<tr>
<td>NCE-5</td>
<td>38</td>
<td>60</td>
<td>45</td>
<td>53</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>Time [min]</th>
<th>Syntactic [%]</th>
<th>Semantic [%]</th>
<th>Total accuracy [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>NEG-5</td>
<td>14</td>
<td>61</td>
<td>58</td>
<td>60</td>
</tr>
<tr>
<td>NEG-15</td>
<td>36</td>
<td>61</td>
<td>61</td>
<td>61</td>
</tr>
<tr>
<td>HS-Huffman</td>
<td>21</td>
<td>52</td>
<td>59</td>
<td>55</td>
</tr>
</tbody>
</table>

The following results use 10^-5 subsampling

Table 1: Accuracy of various Skip-gram 300-dimensional models on the analogical reasoning task as defined in [8]. NEG-k stands for Negative Sampling with k negative samples for each positive sample; NCE stands for Noise Contrastive Estimation and HS-Huffman stands for the Hierarchical Softmax with the frequency-based Huffman codes.
Phrases

Data-driven approach to find the phrases (words that appear frequently together and infrequently in other contexts)

\[
\text{score}(w_i, w_j) = \frac{\text{count}(w_i w_j) - \delta}{\text{count}(w_i) \times \text{count}(w_j)}
\]

\(\delta\) is discounting coefficient

\(\delta\) prevents phrases made of infrequent words
Phrases

Data-driven approach to find the phrases (words that appear frequently together and infrequently in other contexts)

\[
score(w_i, w_j) = \frac{\text{count}(w_iw_j) - \delta}{\text{count}(w_i) \times \text{count}(w_j)}
\]

\(\delta\) is discounting coefficient

\(\delta\) prevents phrases made of infrequent words

Training

2-4 passes over data to form longer sequences
Demo for phrases
Evaluation of ‘Phrases’

**New test set (3218, 5 categories only):**

- Boston Boston_Celtics Miami Miami_Heat
- Werner_Vogels Amazon Samuel_J._Palmisano IBM
- Germany Lufthansa Spain Spanair
- Atlanta Atlanta_Thrashers Boston Boston_Bruins
- Boston Boston_Globe Seattle Seattle_Times
Evaluation of ‘Phrases’ (result)

1 b, dim = 300, context= window-5

<table>
<thead>
<tr>
<th>Method</th>
<th>Dimensionality</th>
<th>No subsampling [%]</th>
<th>$10^{-5}$ subsampling [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>NEG-5</td>
<td>300</td>
<td>24</td>
<td>27</td>
</tr>
<tr>
<td>NEG-15</td>
<td>300</td>
<td>27</td>
<td>42</td>
</tr>
<tr>
<td>HS-Huffman</td>
<td>300</td>
<td>19</td>
<td>47</td>
</tr>
</tbody>
</table>

Table 3: Accuracies of the Skip-gram models on the phrase analogy dataset. The models were trained on approximately one billion words from the news dataset.
Evaluation of ‘Phrases’ (result)

1 b, dim = 300, context = window-5

<table>
<thead>
<tr>
<th>Method</th>
<th>Dimensionality</th>
<th>No subsampling [%]</th>
<th>$10^{-5}$ subsampling [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>NEG-5</td>
<td>300</td>
<td>24</td>
<td>27</td>
</tr>
<tr>
<td>NEG-15</td>
<td>300</td>
<td>27</td>
<td>42</td>
</tr>
<tr>
<td>HS-Huffman</td>
<td>300</td>
<td>19</td>
<td>47</td>
</tr>
</tbody>
</table>

Table 3: Accuracies of the Skip-gram models on the phrase analogy dataset. The models were trained on approximately one billion words from the news dataset.

6 b, dim = 1000, context = sentence -> accuracy 66%

33 b, dim = 1000, context = sentence -> accuracy 72%
Evaluation of ‘Phrases’ (result)

1 b, dim = 300, context = window-5

<table>
<thead>
<tr>
<th>Method</th>
<th>Dimensionality</th>
<th>No subsampling [%]</th>
<th>$10^{-5}$ subsampling [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>NEG-5</td>
<td>300</td>
<td>24</td>
<td>27</td>
</tr>
<tr>
<td>NEG-15</td>
<td>300</td>
<td>27</td>
<td>42</td>
</tr>
<tr>
<td>HS-Huffman</td>
<td>300</td>
<td>19</td>
<td>47</td>
</tr>
</tbody>
</table>

Table 3: Accuracies of the Skip-gram models on the phrase analogy dataset. The models were trained on approximately one billion words from the news dataset.

6 b, dim = 1000, context = sentence -> accuracy 66%

33 b, dim = 1000, context = sentence -> accuracy 72%

Hierarchical softmax and subsampling; amount of data is crucial
Additive compositionality

<table>
<thead>
<tr>
<th>Czech + currency</th>
<th>Vietnam + capital</th>
<th>German + airlines</th>
<th>Russian + river</th>
<th>French + actress</th>
</tr>
</thead>
<tbody>
<tr>
<td>koruna</td>
<td>Hanoi</td>
<td>airline Lufthansa</td>
<td>Moscow</td>
<td>Juliette Binoche</td>
</tr>
<tr>
<td>Check crown</td>
<td>Ho Chi Minh City</td>
<td>carrier Lufthansa</td>
<td>Volga River</td>
<td>Vanessa Paradis</td>
</tr>
<tr>
<td>Polish zolty</td>
<td>Viet Nam</td>
<td>flag carrier Lufthansa</td>
<td>upriver</td>
<td>Charlotte Gainsbourg</td>
</tr>
<tr>
<td>CTK</td>
<td>Vietnamese</td>
<td>Lufthansa</td>
<td>Russia</td>
<td>Cecile De</td>
</tr>
</tbody>
</table>

Table 5: Vector compositionality using element-wise addition. Four closest tokens to the sum of two vectors are shown, using the best Skip-gram model.
Empirical comparison with previous results

<table>
<thead>
<tr>
<th>Model (training time)</th>
<th>Redmond</th>
<th>Havel</th>
<th>ninjutsu</th>
<th>graffiti</th>
<th>capitulate</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>conyers</td>
<td>plauen</td>
<td>reiki</td>
<td>cheesecake</td>
<td>abdicate</td>
</tr>
<tr>
<td>Collobert (50d)</td>
<td>lubbock</td>
<td>dzerzhinsky</td>
<td>kohona</td>
<td>gossip</td>
<td>accede</td>
</tr>
<tr>
<td>(2 months)</td>
<td>keene</td>
<td>osterreich</td>
<td>karate</td>
<td>dioramas</td>
<td>rearm</td>
</tr>
<tr>
<td></td>
<td>McCarthy</td>
<td>Jewell</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Alston</td>
<td>Arzu</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Cousins</td>
<td>Ovitz</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Turian (200d)</td>
<td>Podhurst</td>
<td>Pontiff</td>
<td></td>
<td></td>
<td>anaesthetics</td>
</tr>
<tr>
<td>(few weeks)</td>
<td>Harlang</td>
<td>Pinochet</td>
<td></td>
<td></td>
<td>monkeys</td>
</tr>
<tr>
<td></td>
<td>Agarwal</td>
<td>Rodionov</td>
<td></td>
<td></td>
<td>Jews</td>
</tr>
<tr>
<td>Mnih (100d)</td>
<td>Redmond Wash.</td>
<td>Vaclav Havel</td>
<td>ninja</td>
<td>spray paint</td>
<td></td>
</tr>
<tr>
<td>(7 days)</td>
<td>Redmond Washington</td>
<td>president Vaclav Havel</td>
<td>martial arts</td>
<td>capitation</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Microsoft</td>
<td>Velvet Revolution</td>
<td>swordsmanship</td>
<td>capitated</td>
<td></td>
</tr>
</tbody>
</table>

Table 6: Examples of the closest tokens given various well known models and the Skip-gram model trained on phrases using over 30 billion training words. An empty cell means that the word was not in the vocabulary.
Conclusion

Distributed vector representation can capture a large number of precise syntactic and semantic word relationships.
Conclusion

Distributed vector representation can capture a large number of precise syntactic and semantic word relationships

more regular word representations
improved training speed
new approach called Negative sampling
new approach called Subsampling
Conclusion (in details)

The **hyper-parameter choice** is crucial for **performance** (both speed and accuracy)

The main choices to make are:

**architecture:** skip-gram (slower, better for infrequent words) vs CBOw (fast)

**the training algorithm:**
- **hierarchical softmax** (better for infrequent words)
- vs
- **negative sampling** (better for frequent words, better with low dimensional vectors)

**sub-sampling of frequent words:** can improve both accuracy and speed for large data sets (useful values are in range 1e-3 to 1e-5)

**dimensionality of the word vectors:** usually more is better, but not always

**context (window) size:** for skip-gram usually around 10, for CBOw around 5

[https://code.google.com/p/word2vec/](https://code.google.com/p/word2vec/)
What was after the paper?
What was after the paper?

A lot!
What was after the paper?

A lot!

Names to follow:
Socher, Manning, Omer Levy, Yoav Goldberg…

Why does this produce good word representations? Good question. We don’t really know (Levy, Goldberg, 2014)
Resourceful links

- word2vec Explained: Deriving Mikolov et al.’s Negative-Sampling Word-Embedding Method
- Richard Socher lecture and course
- Hierarchical softmax in neural network language model
- Linguistic Regularities in Sparse and Explicit Word Representations
- Short tutorial about word2vec in Python
- Distributed Representations of Sentences and Documents: Doc2vec(Paragraph2Vec)
Thank you