CS 559 Deep Learning

Word Embeddings and Language Models

Gokberk Cinbis
Word Embeddings
Slide adapted from Antonio Bonafonte
Representation of categorical features

Variable which can take a limited number of possible values
E.g.: gender, blood types, countries, …, letters, words, phonemes

Newsgroup task:
Input: 1000 words (from a fixed vocabulary $V$, size $|V|$)

Phonetics transcription (CMU DICT):
Input: letters: (a b c…. z ‘ - .) (30 symbols)
Example: letters. \(|V| = 30\)

\[
\begin{align*}
\text{‘a’} & : \mathbf{x}^T = [1, 0, 0, \ldots, 0] \\
\text{‘b’} & : \mathbf{x}^T = [0, 1, 0, \ldots, 0] \\
\text{‘c’} & : \mathbf{x}^T = [0, 0, 1, \ldots, 0] \\
\vdots & \\
\text{‘.’} & : \mathbf{x}^T = [0, 0, 0, \ldots, 1]
\end{align*}
\]
One-hot (one-of-n) encoding

Example: words.

\[ x^T \text{cat: } = [1, 0, 0, \ldots, 0] \]
\[ x^T \text{dog: } = [0, 1, 0, \ldots, 0] \]

\[ x^T \text{bird: } = [0, 0, 0, \ldots, 0, 1, 0, \ldots, 0] \]

Slide adapted from Antonio Bonafonte
One-hot (one-of-n) encoding

Example: words.

\[
\begin{align*}
\text{cat: } x^T &= [1, 0, 0, \ldots, 0] \\
\text{dog: } x^T &= [0, 1, 0, \ldots, 0] \\
\text{bird: } x^T &= [0, 0, 0, \ldots, 0, 1, 0, \ldots, 0]
\end{align*}
\]

Number of words, $|V|$?

- B2: 5K
- C2: 18K
- LVSR: 50-100K
- Wikipedia (1.6B): 400K
- Crawl data (42B): 2M
One-hot (one-of-n) encoding

Example: words.

\[
\begin{align*}
\text{cat: } x^T & = [1, 0, 0, \ldots, 0] \\
\text{dog: } x^T & = [0, 1, 0, \ldots, 0] \\
\text{bird: } x^T & = [0, 0, 0, \ldots, 0, 1, 0, \ldots, 0]
\end{align*}
\]

Number of words, $|V|$?

- B2: 5K
- C2: 18K
- LVSR: 50-100K
- Wikipedia (1.6B): 400K
- Crawl data (42B): 2M

$x$ needs to be 2M-dimensional!
One-hot encoding of words: limitations

- Large dimensionality
- Sparse representation (mostly zeros)
- Blind representation
  - Only operators: ‘!=’ and ‘==’
  - ie. there is no inherent notion of similarity

Slide adapted from Antonio Bonafonte
Word embeddings

- Represent words using vectors of dimension $d$ ($\sim 100 - 500$)
- Meaningful (semantic, syntactic) distances
- Dominant research topic in last years in NLP conferences (EMNLP)
- *Good* embeddings are useful for *many* other tasks

Slide adapted from Antonio Bonafonte
How to define word representation?

- JR Firth, 1957: *You shall know a word by the company it keeps.*
- We can use Neural Networks to create word embeddings by modeling local context
PNLM

- PNLM (Bengio et al. JMLR 2003) use a distributed real-valued representation of words and contexts.
- Each word in the vocabulary is mapped to an $m$-dimensional real-valued vector (eg, $m=100$).
PNLM

- **Layer 1:** project V-dimensional one-hot vector to m-dimensions.

- **Layer 2:** transform concatenation of m-dimensional vectors into another vector.

- **Layer 3:** predict next word

---

Slide adapted from Christof Monz
Projection (Layer 1) can be considered as word embedding

Here, the main objective is next-word prediction

Word embedding is a by-product of the LM

i-th output = P(w_t = i | context)
PNLM

Expensive!

Slide adapted from Christof Monz
Word embeddings: requirements

The original PNLM model is too expensive

We need to train efficiently over
● very large lexicon
● huge amounts of unsupervised data

word2vec meets these requirements

Slide adapted from Antonio Bonafonte
word2vec

- Proposed by Tomas Mikolov et al. specifically for learning word embeddings
- Shallow NNs for encoding linguistic context of words
- Comparison to PNLM:
  - Less expensive model (remove expensive hidden layer)
  - Better modeling of context
  - Better training strategies (avoid expensive softmax)
  - Trained over much larger data
Two approaches:

1) Continuous Bag of Words (CBOW)
2) Skip-Gram
CBOW: Continuous Bag of Words

Given context:
\text{a, cat, the, tree}

Estimate prob. of \text{climbed}
CBOW: Continuous Bag of Words

- No nonlinearity
- One-hidden layer (h)
- $h = W_1 \ast w_{context}$
  
  $w_{context}$: sum of 1-hot vectors
- Output: $\text{softmax}(W_2 \ast h)$ where $W_2: |V| \times |h|$
CBOW: Continuous Bag of Words

- We can use columns of $W_1$ (or rows of $W_2$) as word embeddings
- Alternatively, use their average
Skip-gram

Given word: climbed

Estimate prob. of context words: a, cat, the, tree

the cat climbed a tree

Slide adapted from Antonio Bonafonte
But softmax is slow

Softmax over $|V|$ numbers is expensive!

From tensorflow word2vec tutorial: https://www.tensorflow.org/tutorials/word2vec/
But softmax is slow

Softmax over $|V|$ numbers is expensive!

Use hierarchical softmax OR avoid by negative sampling

From tensorflow word2vec tutorial: https://www.tensorflow.org/tutorials/word2vec/
Negative sampling

Just discriminate true word from k noise words instead

Note: There are other alternatives for replacing softmax

From tensorflow word2vec tutorial: [https://www.tensorflow.org/tutorials/word2vec/](https://www.tensorflow.org/tutorials/word2vec/)
Evaluation of Word Embeddings

- **Word similarity tasks**
  - Rank list of word pairs, e.g., \((\text{car}, \text{bicycle})\), by similarity
  - Spearman correlation with human judgements
  - Benchmarks: WS-353, Simlex-999, ...
  - Mixes all kinds of similarities (synonyms, topical, unrelated...)

- **Analogy task**
  - Paris is to France as Berlin is to X
  - Evaluated by accuracy
  - Also includes syntactic analogy: \(\text{acquired}\) is to \(\text{acquire}\) as \(\text{tried}\) is to X
  - Arithmetic magic: \(X = v_{\text{king}} - v_{\text{man}} + v_{\text{woman}}\)
Evaluation of Word Embeddings

Certain directions in word2vec space capture specific semantic relations.

Reference: www.tensorflow.org/tutorials/word2vec
Discussion of word2vec

Word2Vec is not deep, but used in many tasks using deep learning

There are several other approaches, like GloVe (Pennington et al).

Why does it works? See GloVe paper.
We can get embeddings implicitly from any task that involves words.

However, good generic embeddings are good for other tasks which may have much less data (transfer learning).

Sometimes, the embeddings can be fine-tuned to the final task.
Language Models

Slides adapted from Marta Ruiz Costa-jussà
What is the most probable sentence?

Two birds are flying  Two beards are flying
Probability of a sentence

• Suppose you record a database of one billion utterances in English.
• If the sentence “how's it going?” appears 76,413 times in that database, then we say
  \[ P(\text{how's it going?}) = \frac{76,413}{1,000,000,000} \]
A language model finds the probability of a sentence

- Given a sentence \((w_1, w_2, \ldots, w_T)\),
- What is \(p(w_1, w_2, \ldots, w_T) = ?\)
An n-gram language model
Chain rule probability and Markov simplifying assumption

\[ p(w_1, w_2, \ldots, w_T) = p(w_T|w(T-1),w(T-2)\ldots w_1) \cdot p(w(T-1)|w(T-2),w(T-3)\ldots w_1) \cdots p(w_1) \]

Markov simplifying assumption: The current word only depends on \( n \) previous words.

\[ p(w_t|w(t-1)w(t-2)\ldots w_1) \sim p(w_t|w(t-1)) \]
Chain rule probability and Markov simplifying assumption

\[ p(w_1, w_2, \ldots, w_T) = p(w_T|w(T-1), w(T-2), \ldots, w_1) \cdot p(w(T-1)|w(T-2), w(T-3), \ldots, w_1) \cdots \cdot p(w_1) \]

Markov simplifying assumption: The current word only depends on \( n \) previous words.

\[
p(w_t|w(t-1), w(t-2), \ldots, w_1) \approx p(w_t|w(t-1))
\]

Objective
An n-gram-based language model

• An n-word substring is called an n-gram.
• If n=1, we say unigram; if n=2, we say bigram; if n=3, we say trigram.
• \( P(<s> \text{ I like snakes that are not poisonous } </s>) \sim \)
  \( b(I \mid <s>) \cdot b(\text{like} \mid I) \cdot b(\text{snakes} \mid \text{like}) \cdot \ldots \cdot b(\text{poisonous} \mid \text{not}) \cdot b(</s> \mid \text{poisonous}) \)
An n-gram-based language model

Unigram probabilities

\[ p(w_1) = \frac{\text{count}(w_1)}{\text{total words observed}} \]

Bigram probabilities

\[ p(w_2|w_1) = \frac{\text{count}(w_1w_2)}{\text{count}(w_1)} \]

Trigram probabilities

\[ p(w_3|w_1w_2) = \frac{\text{count}(w_1w_2w_3)}{\text{count}(w_1w_2)} \]
An n-gram-based language model

Unigram probabilities

\[ p(w_1) = \frac{\text{count}(w_1)}{\text{total words observed}} \]

Bigram probabilities

\[ p(w_2|w_1) = \frac{\text{count}(w_1w_2)}{\text{count}(w_1)} \]

Trigram probabilities

\[ p(w_3|w_1w_2) = \frac{\text{count}(w_1w_2w_3)}{\text{count}(w_1w_2)} \]

Do you see a source of problem in this model?
Some examples...

"<s> que la fuerza te acompañe </s>", = *may the force be with you*

bigrams and trigrams like:

*fuerza te*

*la fuerza te*

*fuerza te acompañe te acompañe </s>*

do not appear in the big corpus of El Periodico (40 M words)

BUT PROBABILITY OF THE SENTENCE SHOULD NOT BE ZERO!!!!
Sparse counts are a big problem

- Fall back to (n-1)-gram, (n-2)-gram,... so on avoids zero probabilities

\[
.8 \times p(w_3|w_1w_2) \\
+.15 \times p(w_3|w_2) \\
+0.049 \times p(w_3) \\
+.001
\]
Sparse counts are a big problem

- Smoothing avoids zero probabilities

\[ .8 \times p(w_3|w_1w_2) + .15 \times p(w_3|w_2) + 0.049 \times p(w_3) + .001 \]
Lack of generalization is still an issue

Mary buys two apples and two oranges in the market.
three apples are for me

the tree has three oranges
A learned function that takes $n-1$ words as input and returns a conditional probability of the next one.
Recap: neural language model

Q: Why does this model have a better generalization for unseen n-grams?

Figure: K. Cho, DL4MT course, 2015
Recap: neural language model

Q: Why does this model have a better generalization for unseen n-grams?

Answer: word embedding
Further generalization comes from word embeddings

Mary buys two apples and two oranges in the market
three apples are for me
the tree has three oranges
Markov assumption can be a limiting factor

Neural language model still assumes the n-th order Markov property

In France, there are around 66 million people and they speak French.
How we can handle variable-length input?

\[ p(w_1, w_2, \ldots, w_T) = \prod_{t=1}^{T} p(w_t | w_1 \ldots w_{t-1}) \]
How we can handle variable-length input?

\[ p(w_1, w_2, \ldots, w_T) = \prod_{t=1}^{T} p(w_t|w_1 \ldots w_{t-1}) \]

Use a (gated) RNN!

For instance, an LSTM with

- input: one-hot (or pre-trained word-embedding)
- output: softmax over vocabulary
A comparison of language models

Perplexity measures how high a probability the language model assigns to correct next words in the test corpus “on average”. A better language model is the one with a lower perplexity.

<table>
<thead>
<tr>
<th>LM</th>
<th>Hidden Layers</th>
<th>PPL</th>
</tr>
</thead>
<tbody>
<tr>
<td>n-gram-based</td>
<td></td>
<td>131.2</td>
</tr>
<tr>
<td>+feed-forward</td>
<td>600</td>
<td>112.5</td>
</tr>
<tr>
<td>+RNN</td>
<td>600</td>
<td>108.1</td>
</tr>
<tr>
<td>+LSTM</td>
<td>600</td>
<td>92.0</td>
</tr>
</tbody>
</table>

Results from Sundermeyer et al, 2015
Summary

- Language modeling consists in assigning a probability to a sequence of words.
- We can model a sequence of words with n-grams, feed-forward networks and recurrent networks.
- Feed-forward networks are able to generalise unseen contexts.
- RNNs are able to use variable contexts.
References

- Mikolov, Tomas; et al. "Efficient Estimation of Word Representations in Vector Space
- Linguistic Regularities in Continuous Space Word Representations
- Tensorflow tutorial https://www.tensorflow.org/tutorials/word2vec/
- GloVe: http://nlp.stanford.edu/projects/glove
- Blog: sebastianruder.com/word-embeddings-1