Introduction to NLP and Word Embeddings

Danna Gurari
University of Colorado Boulder
Spring 2024

https://dannagurari.colorado.edu/course/neural-networks-and-deep-learning-spring-2024/
Review

• Last week:
  • Machine learning for sequential data
  • Recurrent neural networks (RNNs)
  • Problem: learning challenges
  • Solution: Gated RNNs
  • Programming tutorial

• Assignments (Canvas):
  • Problem set 3 due earlier today
  • Lab assignment 3 (final one!) due in a 1.5 weeks

• Questions?
Today’s Topics

• Introduction to natural language processing

• Text representation

• One hot encodings

• Neural word embeddings

• Programming tutorial
Today’s Topics

• Introduction to natural language processing
• Text representation
• One hot encodings
• Neural word embeddings
• Programming tutorial
NLP: Computers that Can Understand (and So Also Communicate in) Human Language

We will focus today only on textual data.
Why Discuss NLP With RNNs?

• RNNs have a strong track record for NLP problems
• Text data’s representation (i.e., sequential data) is a natural match for RNNs
Historical Context: Origins of NLP

Research community emerged mostly on the problem of translating languages.

- 1847: Gradient descent
- 1945: First programmable machine
- 1950: Turing test
- 1956: AI
- 1959: Perceptron
- Machine learning
- 2012: Wave 3: rise of “deep learning”

NLP in Context

Artificial Intelligence
(machines that do “intelligent” things)

Course scope

Deep Learning

Natural Language Processing

Computer Vision
Key Challenge: Replicate Language Understanding for So Many Tasks!

- Text classification
- Machine translation
- Question answering
- Automatic summarization
- And more...
Key Challenge: Replicate Language Understanding for So Many Tasks!

- Text classification
- Machine translation
- Question answering
- Automatic summarization
- And more...
Key Challenge: Replicate Language Understanding for So Many Tasks!

- Text classification
- Machine translation
- Question answering
- Automatic summarization
- And more…
Key Challenge: Replicate Language Understanding for So Many Tasks!

- Text classification
- Machine translation
- Question answering
- Automatic summarization
- And more...

e.g., Microsoft translator

Key Challenge: Replicate Language Understanding for So Many Tasks!

- Text classification
- Machine translation
- Question answering
- Automatic summarization
- And more...

Key Challenge: Replicate Language Understanding for So Many Tasks!

- Text classification
- Machine translation
- Question answering
- Automatic summarization
- And more...

e.g., IBM Watson question answering system (and Jeopardy winner)

Key Challenge: Replicate Language Understanding for So Many Tasks!

- Text classification
- Machine translation
- Question answering
- Automatic summarization
- And more...

Get a feel for what wordtune read can do!

Whether you are reading a lengthy blog post, analyzing a report or going through a paper, Wordtune helps you digest what's important. Cut through the word count to find the words that count.

[Links to categories]
Key Challenge: Replicate Language Understanding for So Many Tasks!

• Text classification
• Machine translation
• Question answering
• Automatic summarization

• And more...
Other Key Challenges: Replicate Language Understanding for So Many Languages/Individuals!

- Need a computable characterization of all human languages that simultaneously captures nuances from individuals; e.g., 7000+ languages spoken around the world
Today’s Topics

• Introduction to natural language processing

• Text representation

• One hot encodings

• Neural word embeddings

• Programming tutorial
Input: String (Collection of Characters)

- Common terms
  - **Corpus:** dataset
  - **Document:** example

**Machine learning**

From Wikipedia, the free encyclopedia


**Machine learning** is a field of computer science that uses statistical techniques to give computer systems the ability to "learn" (e.g., progressively improve performance on a specific task) with data, without being explicitly programmed.[2]

The name **machine learning** was coined in 1959 by Arthur Samuel.[11] Machine learning explores the study and construction of algorithms that can learn from and make predictions on data[3]—such algorithms overcome following strictly static program instructions by making data-driven predictions or decisions,[4][2] through building a model from sample inputs. Machine learning is employed in a range of computing tasks where designing and programming explicit algorithms with good performance is difficult or infeasible; example applications include email filtering, detection of network intruders, and computer vision.
Input: Which “String” Feature Types Apply?

• Categorical data
  • Comes from a fixed list (e.g., education level)

• Structured string data
  • e.g., addresses, dates, telephone numbers,

• Text data
How to Describe Text to a Computer?

• Challenge: input often varies in length

Machine learning

From Wikipedia, the free encyclopedia

For the journal, see Machine Learning (journal).

"Statistical learning" redirects here. For statistical learning in linguistics, see statistical learning in language.

Machine learning is a field of computer science that uses statistical techniques to give computer systems the ability to "learn" (e.g., progressively improve performance on a specific task) with data, without being explicitly programmed.[2]

The name machine learning was coined in 1959 by Arthur Samuel.[1] Machine learning explores the study and construction of algorithms that can learn from and make predictions on data[3] — such algorithms overcome following strictly static program instructions by making data-driven predictions or decisions,[4] through building a model from sample inputs. Machine learning is employed in a range of computing tasks where designing and programming explicit algorithms with good performance is difficult or infeasible; example applications include email filtering, detection of network intruders, and computer vision.

• Solution: convert text to fixed numeric format
Today’s Topics

• Introduction to natural language processing

• Text representation

• One hot encodings

• Neural word embeddings

• Programming tutorial
Converting Text to Vectors

1. Tokenize training data
2. Learn vocabulary
3. Encode data as vectors
Converting Text to Vectors

1. Tokenize training data; convert data into sequence of tokens (e.g., data -> “This is tokening”)
2. Learn vocabulary
3. Encode data as vectors

Two common approaches:

**Character Level**

```
[This] [i] [s] [i] [s] [t] [o] [k] [e] [n] [i] [z] [i] [n] [g] [.]
```

**Word Level**

```
[This] [i] [s] [t] okenizing [.]
```
Converting Text to Vectors

1. Tokenize training data
2. Learn vocabulary by identifying all unique tokens in the training data
3. Encode data as vectors

Two common approaches:

**Character Level**

<table>
<thead>
<tr>
<th>Token</th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>***</th>
<th>0</th>
<th>1</th>
<th>***</th>
<th>!</th>
<th>@</th>
<th>***</th>
</tr>
</thead>
<tbody>
<tr>
<td>Index</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>***</td>
<td>27</td>
<td>28</td>
<td>***</td>
<td>119</td>
<td>120</td>
<td>***</td>
</tr>
</tbody>
</table>

**Word Level**

<table>
<thead>
<tr>
<th>Token</th>
<th>a</th>
<th>an</th>
<th>at</th>
<th>***</th>
<th>bat</th>
<th>ball</th>
<th>***</th>
<th>zipper</th>
<th>zoo</th>
<th>***</th>
</tr>
</thead>
<tbody>
<tr>
<td>Index</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>***</td>
<td>527</td>
<td>528</td>
<td>***</td>
<td>9,842</td>
<td>9,843</td>
<td>***</td>
</tr>
</tbody>
</table>
Converting Text to Vectors

1. Tokenize training data
2. Learn vocabulary by identifying all unique tokens in the training data
3. Encode data as one-hot vectors
Converting Text to Vectors

What are the pros and cons for using word tokens instead of character tokens?

- **Pros:** length of input/output sequences is shorter, simplifies learning semantics
- **Cons:** “UNK” word token needed for out of vocabulary words; vocabulary can be large
Converting Text to Vectors

<table>
<thead>
<tr>
<th>Character Level</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Token</strong></td>
<td>a</td>
<td>b</td>
<td>c</td>
<td>***</td>
<td>0</td>
<td>1</td>
<td>***</td>
<td>!</td>
<td>@</td>
</tr>
<tr>
<td><strong>Index</strong></td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>***</td>
<td>27</td>
<td>28</td>
<td>***</td>
<td>119</td>
<td>120</td>
</tr>
</tbody>
</table>

Word Level

<table>
<thead>
<tr>
<th><strong>Token</strong></th>
<th>a</th>
<th>an</th>
<th>at</th>
<th>***</th>
<th>bat</th>
<th>ball</th>
<th>***</th>
<th>zipper</th>
<th>zoo</th>
<th>***</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Index</strong></td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>***</td>
<td>527</td>
<td>528</td>
<td>***</td>
<td>9,842</td>
<td>9,843</td>
<td>***</td>
</tr>
</tbody>
</table>

Word level representations are more commonly used
Implementation Details – Possible Pre-processing

• Lower case all letters
• Stemming: use each word’s stem; e.g., singular to plural, resolve different verb forms
  • e.g.,

![Stemming Diagram](https://dzone.com/articles/using-lucene-grails)

• Stop word removal: discard frequent words

[Stop words](https://github.com/topics/stopwords-removal)
Problems with One-Hot Encoding Words?

- Huge memory burden
- Computationally expensive

Dimensionality = vocabulary size

e.g., English has ~170,000 words with ~10,000 commonly used words

Limitation of One-Hot Encoding Words

• No notion of which words are similar, yet such understanding can improve generalization
  • e.g., “walking”, “running”, and “skipping” are all suitable for “He was _____ to school.”

The distance between all words is equal!
Today’s Topics

• Introduction to natural language processing

• Text representation

• One hot encodings

• Neural word embeddings

• Programming tutorial
Idea: Represent Each Word Compactly in a Space Where Vector Distance Indicates Word Similarity

Recap of Big Picture

• Convert words into compact vectors as **input** to neural networks; e.g., RNNs

Inspiration: Distributional Semantics

“The distributional hypothesis says that the meaning of a word is derived from the context in which it is used, and words with similar meaning are used in similar contexts.”

- Origins: Harris in 1954 and Firth in 1957

Inspiration: Distributional Semantics

“The distributional hypothesis says that the meaning of a word is derived from the context in which it is used, and words with similar meaning are used in similar contexts.”

Inspiration: Distributional Semantics

• What is the meaning of berimbau based on context?

  Background music from a berimbau offers a beautiful escape.
  Many people danced around the berimbau player.
  I practiced for many years to learn how to play the berimbau.

• Idea: context makes it easier to understand a word’s meaning

[Adapted from slides by Lena Voita]
Inspiration: Distributional Semantics

“The distributional hypothesis says that the meaning of a word is derived from the context in which it is used, and words with similar meaning are used in similar contexts.”

Inspiration: Distributional Semantics

• What other words could fit into these contexts?

1. Background music from a _______ offers a beautiful escape.
2. Many people danced around the _______ player.
3. I practiced for many years to learn how to play the _______.

Hypothesis is that words with similar row values have similar meanings.
Inspiration: Distributional Semantics

“The distributional hypothesis says that the meaning of a word is derived from the context in which it is used, and words with similar meaning are used in similar contexts.”

Approach

• Learn a dense (lower-dimensional) vector for each word by characterizing its context, which inherently will reflect similarity/differences to other words.
Approach

• Learn a dense (lower-dimensional) vector for each word by characterizing its **context**, which inherently will reflect similarity/differences to other words

We embed words in a shared space so they can be compared with a few features

What features would discriminate these words?
Approach

- Learn a dense (lower-dimensional) vector for each word by characterizing its **context**, which inherently will reflect similarity/differences to other words.
Approach: Learn Word Embedding Space

• An embedding space represents a finite number of words, decided in training
• A word embedding is represented as a vector indicating its context
• The dimensionality of all word embeddings in an embedding space match
  • What is the word embedding dimensionality for the shown example?
Approach: Learn Word Embedding Space

• An embedding space represents a finite number of words, defined in training
• A word embedding is represented as a vector indicating its context
• The dimensionality of all word embeddings in an embedding space match

In practice, the learned discriminating features are hard to interpret
Embedding Matrix

• The embedding matrix converts an input word into a dense vector

Embedding Matrix

• It converts an input word into a dense vector

A word’s embedding can efficiently be extracted when we know the word’s index

Word Embedding Analogous to a CNN Pretrained Feature

- e.g., FC6 layer of AlexNet
Popular Word Embeddings

• Bengio method

• Word2vec (skip-gram model)

• And more...
Historical Context

1847: Gradient descent
1945: First programmable machine
1950: Turing test
1956-1957: AI
1959: Machine learning
1957: Perceptron
1959: Neural networks with effective learning strategy
1980: Neural networks with Backpropagation for CNNs
1986: MNIST, LeNet
1989: Neocognitron
1998: "Deep learning" Wave 3
2003: Bengio method
2012: Word2vec
Popular Word Embeddings

• Bengio method

• Word2vec (skip-gram model)

• And more...
Idea: Learn Word Embeddings That Help Predict Viable Next Words

e.g.,

1. Background music from a _______

2. Many people danced around the _______

3. I practiced for many years to learn how to play the _______

Task: Predict Next Word
Given Previous Ones

e.g.,

1. Background music from a _______
2. Many people danced around the _______
3. I practiced for many years to learn how to play the _______
Task: Predict Next Word Given Previous Ones

e.g., a vocabulary size of 17,000 was used in experiments

What is the dimensionality of the output layer?
- 17,000 (each indexed position indicates probability of a word)

Architecture

Note: the goal is to learn an embedding matrix and, after training, the rest of the neural network can be discarded.
e.g., a vocabulary size of 17,000 was used with embedding sizes of 30, 60, and 100 in experiments.

Assume a 30-d word embedding - what are the dimensions of the embedding matrix $C$?

$30 \times 17,000$ (i.e., 510,000 weights)
e.g., a vocabulary size of 17,000 was used with embedding sizes of 30, 60, and 100 in experiments.

Assume a 30-d word embedding - what are the dimensions of each word embedding?

1 x 30
Architecture

Projection layer followed by a hidden layer with non-linearity

\[ i\text{-th output} = P(w_t = i \mid \text{context}) \]

softmax

most computation here

tanh

\[ C(w_{t-2}), C(w_{t-1}) \]

Table look-up in \( C \)

Matrix \( C \)

shared parameters across words

index for \( w_{t-n+1} \)

index for \( w_{t-2} \)

index for \( w_{t-1} \)

Training

Use sliding window on input data; e.g., 3 words

Background music from a berimbau offers a beautiful escape...

Input: tried 1, 3, 5, and 8 input words and used 2 datasets with ~1 million and ~34 million words respectively

Training

Use sliding window on input data; e.g., 3 words

Background music from a berimbau offers a beautiful escape...

Input: tried 1, 3, 5, and 8 input words and used 2 datasets with ~1 million and ~34 million words respectively

\[ i \text{-th output} = P(w_t = i \mid \text{context}) \]

\( C(w_{t-n+1}) \)

Table look-up in \( C \)

Index for \( w_{t-n+1} \)

\( C(w_{t-2}) \)

Matrix \( C \)

Shared parameters across words

Index for \( w_{t-2} \)

Index for \( w_{t-1} \)

Training

Use sliding window on input data; e.g., 3 words

Background music from a berimbau offers a beautiful escape...

Input: tried 1, 3, 5, and 8 input words and used 2 datasets with \(~1\) million and \(~34\) million words respectively

\[
i\text{-th output} = P(w_i=i | \text{context})
\]

Training

Use sliding window on input data; e.g., 3 words

Background music from a berimbau offers a beautiful escape...

Input: tried 1, 3, 5, and 8 input words and used 2 datasets with ~1 million and ~34 million words respectively

\[ i\text{-th output} = P(w_t = i \mid \text{context}) \]
Training

Input: tried 1, 3, 5, and 8 input words and used 2 datasets with ~1 million and ~34 million words respectively.

Word embedding iteratively updated

Cost function: minimize cross entropy loss plus regularization (i.e., L2 norm penalty)

\[ i\text{-th output} = P(w_i = i \mid \text{context}) \]

\[ C(w_{t-n+1}) \]

Table look-up in \( C \)

Index for \( w_{t-n+1} \)

\[ C(w_{t-2}) \]

Matrix \( C \)

Shared parameters across words

Index for \( w_{t-2} \)

\[ C(w_{t-1}) \]

Index for \( w_{t-1} \)

Summary: Word Embeddings Learn Context of Previous Words Needed to Predict Next Word

e.g.,

1. Background music from a _______
2. Many people danced around the _______
3. I practiced for many years to learn how to play the _______
Popular Word Embeddings

• Bengio method

• Word2vec (skip-gram model)

• And more...
Idea: Learn Word Embeddings That Know What Are Viable Surrounding Words

e.g.,

1. ___ ___ ___ ___ berimbau ___ ___ ___ ___ ___

2. ___ berimbau ___

Task: Given Word, Predict a Nearby Word

e.g.,

1. ___ ___ ___ ___ berimbau ___ ___ ___ ___ ___
2. ___ berimbau ___
Task: Given Word, Predict a Nearby Word

https://towardsdatascience.com/word2vec-skip-gram-model-part-1-intuition-78614e4d6e0b
Note: the goal is to learn an embedding matrix and, after training, the rest of the neural network can be discarded.

https://towardsdatascience.com/word2vec-skip-gram-model-part-1-intuition-78614e4d6e0b
e.g., a vocabulary size of 10,000 is used with embedding sizes of 300

What are the dimensions of the embedding matrix?

300 x 10,000 (i.e., 3,000,000 weights)
Architecture

e.g., a vocabulary size of 10,000 is used with embedding sizes of 300

What are the dimensions of each word embedding?

1 x 300

https://towardsdatascience.com/word2vec-skip-gram-model-part-1-intuition-78614e4d6e0b
A shallower, simpler architecture than the Bengio approach (i.e., lacks a non-linear hidden layer)!
Training

Sliding window run on input data to sample neighbors of each target word (e.g., using window size of 2)

Source Text

The quick brown fox jumps over the lazy dog.

Training Samples

(the, quick)
(the, brown)
(quick, the)
(quick, brown)
(quick, fox)
(brown, the)
(brown, quick)
(brown, fox)
(brown, jumps)
(fox, quick)
(fox, brown)
(fox, jumps)
(fox, over)
Hyperparameters: What Works Well?

- Word embedding dimensionality?
  - Dimensionality set between 100 and 1,000

- Context window size?
  - ~10

Very Exciting/Surprising Finding

• Vector arithmetic with word embeddings can solve many analogies

  *(Full test list: http://download.tensorflow.org/data/questions-words.txt)*

• **Semantic** relationships (meaning of words in a sentence):
  • Italy + (Paris - France) = Rome

• **Syntactic** relationships (rules for words in a sentence)
  • smallest + (big – small) = biggest
  • think + (read – reading) = thinking
  • mouse + (dollars – dollar) = mice
Summary: Word Embeddings Are Learned that Support Predicting Viable Surrounding Words!

e.g.,

1. ___ ___ ___ ___ berimbau ___ ___ ___ ___

2. ___ berimbau ___
Popular Word Embeddings

• Bengio method

• Word2vec (skip-gram model)

• And more...
Variants for Learning Word Embeddings

• Capture global context rather than just local context of previous or surrounding words; e.g.,
  • GloVe for Global Vectors (Pennington et al., 2014)

• Capture that the same word can have different word vectors under different contexts; e.g.,
  • Elmo for embeddings from language models (Peters et al., arXiv 2018)

• Support multiple languages; e.g.,
  • Fast-text (Bojanowski et al., 2016)
Popular Word Embeddings

• Bengio method

• Word2vec (skip-gram model)

• And more...
Recap of Big Picture

• Convert words into compact vectors as input to neural networks; e.g., RNNs

• Implementation detail: may need to learn extra tokens such as “UNK” and “EOS” to represent out of vocabulary words and signify end of the string respectively

• Also, can fine-tune word embedding matrices for different applications

Word Embedding Limitations/Challenges

• Distinguish antonyms from synonyms
  • Antonyms are learned near each other in the embedding space since they are commonly used in similar contexts: “I hate math” vs “I love math” or “Take a right turn” vs “Take a left turn”

• Gender bias:

<table>
<thead>
<tr>
<th>Man is to Computer Programmer as Woman is to Homemaker? Debiasing Word Embeddings</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tolga Bolukbasi(^1), Kai-Wei Chang(^2), James Zou(^2), Venkatesh Saligrama(^{1,2}), Adam Kalai(^2)</td>
</tr>
<tr>
<td>(^1)Boston University, 8 Saint Mary’s Street, Boston, MA</td>
</tr>
<tr>
<td>(^2)Microsoft Research New England, 1 Memorial Drive, Cambridge, MA</td>
</tr>
<tr>
<td><a href="mailto:tolgab@bu.edu">tolgab@bu.edu</a>, <a href="mailto:kw@kwchang.net">kw@kwchang.net</a>, <a href="mailto:jamesyzou@gmail.com">jamesyzou@gmail.com</a>, <a href="mailto:srv@bu.edu">srv@bu.edu</a>, <a href="mailto:adam.kalai@microsoft.com">adam.kalai@microsoft.com</a></td>
</tr>
</tbody>
</table>
Word Embedding Limitations/Challenges

- Distinguish antonyms from synonyms
  - Antonyms are learned near each other in the embedding space since they are commonly used in similar contexts: “I hate math” vs “I love math” or “Take a right turn” vs “Take a left turn”

- Gender bias:

<table>
<thead>
<tr>
<th>Extreme she</th>
<th>Extreme he</th>
<th>Gender stereotype she-he analogies</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. homemaker</td>
<td>1. maestro</td>
<td>registered nurse-physician</td>
</tr>
<tr>
<td>2. nurse</td>
<td>2. skipper</td>
<td>interior designer-architect</td>
</tr>
<tr>
<td>3. receptionist</td>
<td>3. protege</td>
<td>feminism-conservatism</td>
</tr>
<tr>
<td>4. librarian</td>
<td>4. philosopher</td>
<td>vocalist-guitarist</td>
</tr>
<tr>
<td>5. socialite</td>
<td>5. captain</td>
<td>diva-superstar</td>
</tr>
<tr>
<td>6. hairdresser</td>
<td>6. architect</td>
<td>volleyball-football</td>
</tr>
<tr>
<td>7. nanny</td>
<td>7. financier</td>
<td>cupcakes-pizzas</td>
</tr>
<tr>
<td>8. bookkeeper</td>
<td>8. warrior</td>
<td>housewife-shopkeeper</td>
</tr>
<tr>
<td>9. stylist</td>
<td>9. broadcaster</td>
<td>softball-baseball</td>
</tr>
<tr>
<td>10. housekeeper</td>
<td>10. magician</td>
<td>cosmetics-pharmaceuticals</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Gender appropriate she-he analogies</th>
</tr>
</thead>
<tbody>
<tr>
<td>sister-brother</td>
</tr>
<tr>
<td>mother-father</td>
</tr>
<tr>
<td>ovarian cancer-prostate cancer convent-monastery</td>
</tr>
</tbody>
</table>

Word Embedding Limitations/Challenges

- Distinguish antonyms from synonyms
  - Antonyms are learned near each other in the embedding space since they are commonly used in similar contexts: “I hate math” vs “I love math” or “Take a right turn” vs “Take a left turn”

- Gender bias

- What other language biases do you think could be learned?

Today’s Topics

• Introduction to natural language processing

• Text representation

• One hot encodings

• Neural word embeddings

• Programming tutorial
Today’s Topics

• Introduction to natural language processing

• Text representation

• One hot encodings

• Neural word embeddings

• Programming tutorial
The End