Text Analysis using Neural Networks

Michael Peress

SUNY-Stony Brook

March 31, 2026

Text Analysis using Neural Networks

- When we covered text analysis, we relied on features built on counting the occurrences or frequencies of words (or phrases) in a document:
  - Bag-of-words
  - Bi-grams
  - n-grams
  - Skip-grams

Text Analysis using Neural Networks

- When we covered text analysis, we relied on features built on counting the occurrences or frequencies of words (or phrases) in a document:
  - Bag-of-words
  - Bi-grams
  - n-grams
  - Skip-grams
- Let's attempt to use a neural network with BOW features and compare to Naive Bayes and the LASSO

Text Analysis using Neural Networks

- When we covered text analysis, we relied on features built on counting the occurrences or frequencies of words (or phrases) in a document:
  - Bag-of-words
  - Bi-grams
  - n-grams
  - Skip-grams
- Let's attempt to use a neural network with BOW features and compare to Naive Bayes and the LASSO
- We will consider neural networks with 0 hidden layers, 1 hidden layer, and 2 hidden layers (the 0 hidden layer model is like the LASSO, but using hampered optimization for regularization) with 200 neurons in each hidden layer

Deep Learning for Text Analysis

- Application 1: Sentiment in IMDB Reviews

Deep Learning for Text Analysis

- Application 1: Sentiment in IMDB Reviews
  - 50,000 reviews, with an average length of 231 words, with reviews labeled as positive and negative

Deep Learning for Text Analysis

- Application 1: Sentiment in IMDB Reviews
  - 50,000 reviews, with an average length of 231 words, with reviews labeled as positive and negative
  - A standard teaching dataset that has been thoroughly cleaned (and thus somewhat artificial)

Deep Learning for Text Analysis

- Application 1: Sentiment in IMDB Reviews
  - 50,000 reviews, with an average length of 231 words, with reviews labeled as positive and negative
  - A standard teaching dataset that has been thoroughly cleaned (and thus somewhat artificial)

	Mode	Per. Mode	Y=0	Y=1	All
Naive Bayes, BOW:					
Train	0	50.1	83.8	85.2	84.5
Test	1	50.1	83.2	84.6	83.9
LASSO, BOW:					
Train	0	50.1	88.1	89.8	89.0
Test	1	50.1	85.4	86.6	86.0

Deep Learning for Text Analysis

- Application 1: Sentiment in IMDB Reviews
  - 50,000 reviews, with an average length of 231 words, with reviews labeled as positive and negative
  - A standard teaching dataset that has been thoroughly cleaned (and thus somewhat artificial)

	Mode	Per. Mode	Y=0	Y=1	All
Naive Bayes, BOW:					
Train	0	50.1	83.8	85.2	84.5
Test	1	50.1	83.2	84.6	83.9
LASSO, BOW:					
Train	0	50.1	88.1	89.8	89.0
Test	1	50.1	85.4	86.6	86.0

- Naive Bayes and LASSO perform fairly well

Deep Learning for Text Analysis

- Application 1: Sentiment in IMDB Reviews

	Mode	Per. Mode	Y=0	Y=1	All
Naive Bayes, BOW:					
Train	0	50.1	83.8	85.2	84.5
Test	1	50.1	83.2	84.6	83.9
LASSO, BOW:					
Train	0	50.1	88.1	89.8	89.0
Test	1	50.1	85.4	86.6	86.0
NN, 0 Hidden Layers:					
Train	0	50.1	87.2	90.6	88.9
Test	1	50.1	85.1	87.8	86.5
NN, 1 Hidden Layer:					
Train	0	50.1	99.8	99.8	99.8
Test	1	50.1	86.5	84.3	85.4
NN, 2 Hidden Layers:					
Train	0	50.1	99.9	99.4	99.6
Test	1	50.1	87.0	81.6	84.3

Deep Learning for Text Analysis

- Application 1: Sentiment in IMDB Reviews

	Mode	Per. Mode	Y=0	Y=1	All
Naive Bayes, BOW:					
Train	0	50.1	83.8	85.2	84.5
Test	1	50.1	83.2	84.6	83.9
LASSO, BOW:					
Train	0	50.1	88.1	89.8	89.0
Test	1	50.1	85.4	86.6	86.0
NN, 0 Hidden Layers:					
Train	0	50.1	87.2	90.6	88.9
Test	1	50.1	85.1	87.8	86.5
NN, 1 Hidden Layer:					
Train	0	50.1	99.8	99.8	99.8
Test	1	50.1	86.5	84.3	85.4
NN, 2 Hidden Layers:					
Train	0	50.1	99.9	99.4	99.6
Test	1	50.1	87.0	81.6	84.3

- Neural network with 0 hidden layers performs marginally better than LASSO

Deep Learning for Text Analysis

- Application 1: Sentiment in IMDB Reviews

	Mode	Per. Mode	Y=0	Y=1	All
Naive Bayes, BOW:					
Train	0	50.1	83.8	85.2	84.5
Test	1	50.1	83.2	84.6	83.9
LASSO, BOW:					
Train	0	50.1	88.1	89.8	89.0
Test	1	50.1	85.4	86.6	86.0
NN, 0 Hidden Layers:					
Train	0	50.1	87.2	90.6	88.9
Test	1	50.1	85.1	87.8	86.5
NN, 1 Hidden Layer:					
Train	0	50.1	99.8	99.8	99.8
Test	1	50.1	86.5	84.3	85.4
NN, 2 Hidden Layers:					
Train	0	50.1	99.9	99.4	99.6
Test	1	50.1	87.0	81.6	84.3

- Neural network with 0 hidden layers performs marginally better than LASSO
- Neural network with 1 and 2 hidden layers perform worse than LASSO and seems to be over-fitting

Deep Learning for Text Analysis

- Let's count parameters to understand the over-fitting:
  - The vocabulary size is $W = 1,534$, the number of neurons is $K = 200$, and the number of outcomes is $J = 2$

Deep Learning for Text Analysis

- Let's count parameters to understand the over-fitting:
  - The vocabulary size is $W = 1,534$, the number of neurons is $K = 200$, and the number of outcomes is $J = 2$
  - LASSO has $W + 1 = 1,535$ parameters

Deep Learning for Text Analysis

- Let's count parameters to understand the over-fitting:
  - The vocabulary size is $W = 1,534$, the number of neurons is $K = 200$, and the number of outcomes is $J = 2$
  - LASSO has $W + 1 = 1,535$ parameters
  - Neural network with 0 hidden layers has $(W + 1)J = 3,070$ parameters

Deep Learning for Text Analysis

- Let's count parameters to understand the over-fitting:
  - The vocabulary size is $W = 1,534$, the number of neurons is $K = 200$, and the number of outcomes is $J = 2$
  - LASSO has $W + 1 = 1,535$ parameters
  - Neural network with 0 hidden layers has $(W + 1)J = 3,070$ parameters
  - Neural network with 1 hidden layer has $(W + 1)K + (K + 1)J = 307,402$ parameters

Deep Learning for Text Analysis

- Let's count parameters to understand the over-fitting:
  - The vocabulary size is $W = 1,534$, the number of neurons is $K = 200$, and the number of outcomes is $J = 2$
  - LASSO has $W + 1 = 1,535$ parameters
  - Neural network with 0 hidden layers has $(W + 1)J = 3,070$ parameters
  - Neural network with 1 hidden layer has $(W + 1)K + (K + 1)J = 307,402$ parameters
  - Neural network with 2 hidden layers has $(W + 1)K + (K + 1)K + (K + 1)J = 347,602$ parameters

Deep Learning for Text Analysis

- Let's count parameters to understand the over-fitting:
  - The vocabulary size is $W = 1,534$, the number of neurons is $K = 200$, and the number of outcomes is $J = 2$
  - LASSO has $W + 1 = 1,535$ parameters
  - Neural network with 0 hidden layers has $(W + 1)J = 3,070$ parameters
  - Neural network with 1 hidden layer has $(W + 1)K + (K + 1)J = 307,402$ parameters
  - Neural network with 2 hidden layers has $(W + 1)K + (K + 1)K + (K + 1)J = 347,602$ parameters
  - The mapping between the vocabulary and neurons tends to introduce a large number of parameters—with limited data, this can lead to over-fitting

Deep Learning for Text Analysis

- Let's count parameters to understand the over-fitting:
  - The vocabulary size is $W = 1,534$, the number of neurons is $K = 200$, and the number of outcomes is $J = 2$
  - LASSO has $W + 1 = 1,535$ parameters
  - Neural network with 0 hidden layers has $(W + 1)J = 3,070$ parameters
  - Neural network with 1 hidden layer has $(W + 1)K + (K + 1)J = 307,402$ parameters
  - Neural network with 2 hidden layers has $(W + 1)K + (K + 1)K + (K + 1)J = 347,602$ parameters
  - The mapping between the vocabulary and neurons tends to introduce a large number of parameters—with limited data, this can lead to over-fitting
- Let's consider some other applications before jumping to conclusions

Deep Learning for Text Analysis

- Application 2: Sentiment in Rotten Tomatoes Reviews

Deep Learning for Text Analysis

- Application 2: Sentiment in Rotten Tomatoes Reviews
  - 33,184 reviews, with an average length of 880 words, with reviews labeled as positive or negative by Rotten Tomatoes

Deep Learning for Text Analysis

- Application 2: Sentiment in Rotten Tomatoes Reviews
  - 33,184 reviews, with an average length of 880 words, with reviews labeled as positive or negative by Rotten Tomatoes
  - Reviews are scrapped from links collected by Rotten Tomatoes—the data are dirty (i.e. some html/javascript was not properly removed, some of the 'reviews' are landing pages that indicate a dead link, etc.)

Deep Learning for Text Analysis

- Application 2: Sentiment in Rotten Tomatoes Reviews
  - 33,184 reviews, with an average length of 880 words, with reviews labeled as positive or negative by Rotten Tomatoes
  - Reviews are scrapped from links collected by Rotten Tomatoes—the data are dirty (i.e. some html/javascript was not properly removed, some of the 'reviews' are landing pages that indicate a dead link, etc.)

	Mode	Per. Mode	Y=0	Y=1	All
Naive Bayes, BOW:					
Train	1	69.6	66.8	79.4	75.5
Test	1	68.4	62.9	79.3	74.2
LASSO, BOW:					
Train	1	69.6	77.4	94.7	89.5
Test	1	68.4	60.4	84.8	77.1

Deep Learning for Text Analysis

- Application 2: Sentiment in Rotten Tomatoes Reviews

	Mode	Per. Mode	Y=0	Y=1	All
Naive Bayes, BOW:					
Train	1	69.6	66.8	79.4	75.5
Test	1	68.4	62.9	79.3	74.2
LASSO, BOW:					
Train	1	69.6	77.4	94.7	89.5
Test	1	68.4	60.4	84.8	77.1
NN, 0 Hidden Layers:					
Train	1	69.6	72.1	94.0	87.4
Test	1	68.4	59.4	88.1	79.1
NN, 1 Hidden Layer:					
Train	1	69.6	95.9	97.9	97.3
Test	1	68.4	60.0	86.9	78.4
NN, 2 Hidden Layers:					
Train	1	69.6	91.7	99.5	97.1
Test	1	68.4	56.8	88.2	78.3

Deep Learning for Text Analysis

- Application 2: Sentiment in Rotten Tomatoes Reviews

	Mode	Per. Mode	Y=0	Y=1	All
Naive Bayes, BOW:					
Train	1	69.6	66.8	79.4	75.5
Test	1	68.4	62.9	79.3	74.2
LASSO, BOW:					
Train	1	69.6	77.4	94.7	89.5
Test	1	68.4	60.4	84.8	77.1
NN, 0 Hidden Layers:					
Train	1	69.6	72.1	94.0	87.4
Test	1	68.4	59.4	88.1	79.1
NN, 1 Hidden Layer:					
Train	1	69.6	95.9	97.9	97.3
Test	1	68.4	60.0	86.9	78.4
NN, 2 Hidden Layers:					
Train	1	69.6	91.7	99.5	97.1
Test	1	68.4	56.8	88.2	78.3

- Neural networks seem to perform somewhat better than Naive Bayes and LASSO, though the gains seem to be coming from the type of regularization rather than the nonlinearity (i.e. 0 hidden layers has best fit)

Deep Learning for Text Analysis

- Application 3: Coding whether newspaper articles are discussing the economy (Kayser and Peress, 2021)

Deep Learning for Text Analysis

- Application 3: Coding whether newspaper articles are discussing the economy (Kayser and Peress, 2021)
  - 1,086 newspaper articles with an average length of 690 words in English, French, and German, coded based on whether they are discussing the economy ($N = 1,937$ due to articles being coded by multiple coders)

Deep Learning for Text Analysis

- Application 3: Coding whether newspaper articles are discussing the economy (Kayser and Peress, 2021)
  - 1,086 newspaper articles with an average length of 690 words in English, French, and German, coded based on whether they are discussing the economy ($N = 1,937$ due to articles being coded by multiple coders)

	Mode	Per. Mode	Y=0	Y=1	All
Naive Bayes, BOW:					
Train	1	58.6	78.1	85.9	82.7
Test	1	58.6	72.1	76.2	74.5
LASSO, BOW:					
Train	1	58.6	89.9	94.3	92.5
Test	1	58.6	64.6	72.0	68.9

Deep Learning for Text Analysis

- Application 3: Coding whether newspaper articles are discussing the economy

	Mode	Per. Mode	Y=0	Y=1	All
Naive Bayes, BOW:					
Train	1	58.6	78.1	85.9	82.7
Test	1	58.6	72.1	76.2	74.5
LASSO, BOW:					
Train	1	58.6	89.9	94.3	92.5
Test	1	58.6	64.6	72.0	68.9
NN, 0 Hidden Layers:					
Train	1	58.6	89.3	93.0	91.5
Test	1	58.6	65.9	75.7	71.6
NN, 1 Hidden Layer:					
Train	1	58.6	95.2	90.5	92.4
Test	1	58.6	68.5	72.0	70.6
NN, 2 Hidden Layers:					
Train	1.0	58.6	83.9	98.2	92.3
Test	1.0	58.6	56.1	80.6	70.4

Deep Learning for Text Analysis

- Application 3: Coding whether newspaper articles are discussing the economy

	Mode	Per. Mode	Y=0	Y=1	All
Naive Bayes, BOW:					
Train	1	58.6	78.1	85.9	82.7
Test	1	58.6	72.1	76.2	74.5
LASSO, BOW:					
Train	1	58.6	89.9	94.3	92.5
Test	1	58.6	64.6	72.0	68.9
NN, 0 Hidden Layers:					
Train	1	58.6	89.3	93.0	91.5
Test	1	58.6	65.9	75.7	71.6
NN, 1 Hidden Layer:					
Train	1	58.6	95.2	90.5	92.4
Test	1	58.6	68.5	72.0	70.6
NN, 2 Hidden Layers:					
Train	1.0	58.6	83.9	98.2	92.3
Test	1.0	58.6	56.1	80.6	70.4

- Naive Bayes beats neural networks in this application

Deep Learning for Text Analysis

- Application 3: Coding whether newspaper articles are discussing the economy

	Mode	Per. Mode	Y=0	Y=1	All
Naive Bayes, BOW:					
Train	1	58.6	78.1	85.9	82.7
Test	1	58.6	72.1	76.2	74.5
LASSO, BOW:					
Train	1	58.6	89.9	94.3	92.5
Test	1	58.6	64.6	72.0	68.9
NN, 0 Hidden Layers:					
Train	1	58.6	89.3	93.0	91.5
Test	1	58.6	65.9	75.7	71.6
NN, 1 Hidden Layer:					
Train	1	58.6	95.2	90.5	92.4
Test	1	58.6	68.5	72.0	70.6
NN, 2 Hidden Layers:					
Train	1.0	58.6	83.9	98.2	92.3
Test	1.0	58.6	56.1	80.6	70.4

- Naive Bayes beats neural networks in this application
- Overall, neural networks occasionally lead to modest gains in out of sample fit, with the gains not due to interactions between word sums

Text Analysis using Neural Networks

Drawbacks of Neural Networks with BOW for Text Analysis:

- Bag of words tends to produce a very large number of features, making it harder to apply methods like deep learning which allow for complicated interactions between features

Text Analysis using Neural Networks

Drawbacks of Neural Networks with BOW for Text Analysis:

- Bag of words tends to produce a very large number of features, making it harder to apply methods like deep learning which allow for complicated interactions between features
  - Using occurrences or frequencies of words ignores the fact the synonyms are similar

Text Analysis using Neural Networks

Drawbacks of Neural Networks with BOW for Text Analysis:

- Bag of words tends to produce a very large number of features, making it harder to apply methods like deep learning which allow for complicated interactions between features
  - Using occurrences or frequencies of words ignores the fact the synonyms are similar
  - I'm over-simplifying a bit here—stemming does this to an extent

Text Analysis using Neural Networks

Drawbacks of Neural Networks with BOW for Text Analysis:

- Bag of words tends to produce a very large number of features, making it harder to apply methods like deep learning which allow for complicated interactions between features
  - Using occurrences or frequencies of words ignores the fact the synonyms are similar
  - I'm over-simplifying a bit here—stemming does this to an extent
  - More generally, some words may carry similar meaning without being synonyms (e.g. Albany and Syracuse)

Text Analysis using Neural Networks

Drawbacks of Neural Networks with BOW for Text Analysis:

- Bag of words tends to produce a very large number of features, making it harder to apply methods like deep learning which allow for complicated interactions between features
  - Using occurrences or frequencies of words ignores the fact the synonyms are similar
  - I'm over-simplifying a bit here—stemming does this to an extent
  - More generally, some words may carry similar meaning without being synonyms (e.g. Albany and Syracuse)
- Even with many parameters, the neural network is not capturing the locations of words and the relationships between individual words

Text Analysis using Neural Networks

Drawbacks of Neural Networks with BOW for Text Analysis:

- Bag of words tends to produce a very large number of features, making it harder to apply methods like deep learning which allow for complicated interactions between features
  - Using occurrences or frequencies of words ignores the fact the synonyms are similar
  - I'm over-simplifying a bit here—stemming does this to an extent
  - More generally, some words may carry similar meaning without being synonyms (e.g. Albany and Syracuse)
- Even with many parameters, the neural network is not capturing the locations of words and the relationships between individual words
  - Again, I'm over-simplifying since bi-gram, n-grams, and skip-grams use these to a degree

Text Analysis using Neural Networks

Drawbacks of Neural Networks with BOW for Text Analysis:

- We would like to reduce the number of features and use the locations of words and the relationships between words

Text Analysis using Neural Networks

Drawbacks of Neural Networks with BOW for Text Analysis:

- We would like to reduce the number of features and use the locations of words and the relationships between words
- We first consider an approach for reducing the number of parameters

Text Analysis using Neural Networks

Word Embeddings

- A word may be encoded as text, e.g. “bus”, “car”, “Bolivia”

Text Analysis using Neural Networks

Word Embeddings

- A word may be encoded as text, e.g. “bus”, “car”, “Bolivia”
- The text may be encoded as a number:

Text Analysis using Neural Networks

Word Embeddings

- A word may be encoded as text, e.g. "bus", "car", "Bolivia"
- The text may be encoded as a number:
  - Let $w \in \{1, ..., W\}$ represent a numeric encoding of words with vocabulary size W

Text Analysis using Neural Networks

Word Embeddings

- A word may be encoded as text, e.g. "bus", "car", "Bolivia"
- The text may be encoded as a number:
  - Let $w \in \{1, ..., W\}$ represent a numeric encoding of words with vocabulary size W
  - This type of encoding is unsuitable for modeling because words w and $w + 1$ are not related in any systematic way

Text Analysis using Neural Networks

Word Embeddings

- A word may be encoded as text, e.g. "bus", "car", "Bolivia"
- The text may be encoded as a number:
  - Let $w \in \{1, ..., W\}$ represent a numeric encoding of words with vocabulary size W
  - This type of encoding is unsuitable for modeling because words w and $w + 1$ are not related in any systematic way
- We can represent words using one hot encoding, generating a dummy variable for each value that w can take

Text Analysis using Neural Networks

Word Embeddings

- A word may be encoded as text, e.g. "bus", "car", "Bolivia"
- The text may be encoded as a number:
  - Let $w \in \{1, ..., W\}$ represent a numeric encoding of words with vocabulary size W
  - This type of encoding is unsuitable for modeling because words w and $w + 1$ are not related in any systematic way
- We can represent words using one hot encoding, generating a dummy variable for each value that w can take
  - Bag of words implicity is a sum (or average) of the one hot encoding over words in the document

Text Analysis using Neural Networks

Word Embeddings

- A word may be encoded as text, e.g. "bus", "car", "Bolivia"
- The text may be encoded as a number:
  - Let $w \in \{1, ..., W\}$ represent a numeric encoding of words with vocabulary size W
  - This type of encoding is unsuitable for modeling because words w and $w + 1$ are not related in any systematic way
- We can represent words using one hot encoding, generating a dummy variable for each value that w can take
  - Bag of words implicity is a sum (or average) of the one hot encoding over words in the document
- A word embedding is a representation of words that assigns a real vector value $v \in \mathbb{R}^V$ to a word $w \in \{1, ..., W\}$

Text Analysis using Neural Networks

Word Embeddings

- A word may be encoded as text, e.g. "bus", "car", "Bolivia"
- The text may be encoded as a number:
  - Let $w \in \{1, ..., W\}$ represent a numeric encoding of words with vocabulary size W
  - This type of encoding is unsuitable for modeling because words w and $w + 1$ are not related in any systematic way
- We can represent words using one hot encoding, generating a dummy variable for each value that w can take
  - Bag of words implicity is a sum (or average) of the one hot encoding over words in the document
- A word embedding is a representation of words that assigns a real vector value $v \in \mathbb{R}^V$ to a word $w \in \{1, ..., W\}$
  - Typically, words that are similar in meaning will have close values of v

Text Analysis using Neural Networks

Word Embeddings

- A word may be encoded as text, e.g. "bus", "car", "Bolivia"
- The text may be encoded as a number:
  - Let $w \in \{1, ..., W\}$ represent a numeric encoding of words with vocabulary size W
  - This type of encoding is unsuitable for modeling because words w and $w + 1$ are not related in any systematic way
- We can represent words using one hot encoding, generating a dummy variable for each value that w can take
  - Bag of words implicity is a sum (or average) of the one hot encoding over words in the document
- A word embedding is a representation of words that assigns a real vector value $v \in \mathbb{R}^V$ to a word $w \in \{1, ..., W\}$
  - Typically, words that are similar in meaning will have close values of v
  - Typically, the dimensionality of the representation V will be much smaller than the vocabulary size W

Text Analysis using Neural Networks

Word Embeddings

- Here, we will consider approaches for generating word embeddings—examples include word2vec and GloVe

Text Analysis using Neural Networks

Word Embeddings

- Here, we will consider approaches for generating word embeddings—examples include word2vec and GloVe
- We will see how applying word embeddings allows for text analysis models that are sophisticated in other ways

Text Analysis using Neural Networks

Word Embeddings

- Here, we will consider approaches for generating word embeddings—examples include word2vec and GloVe
- We will see how applying word embeddings allows for text analysis models that are sophisticated in other ways
- We will consider methods that use the macro-structure of the document, such as Recurrent Neural Networks and Transformer models

Word Embeddings

Basic Idea

- Word embeddings are a form of unsupervised learning, in that they require only the independent variables (or features) X and not labeled data

Word Embeddings

Basic Idea

- Word embeddings are a form of unsupervised learning, in that they require only the independent variables (or features) X and not labeled data
- Consider first one hot encodings:
  - We could represent the word "bus" as,

$$(0, ...0, \underset{bus}{1}, 0, ..., 0)$$

Word Embeddings

Basic Idea

- Word embeddings are a form of unsupervised learning, in that they require only the independent variables (or features) X and not labeled data
- Consider first one hot encodings:
  - We could represent the word "bus" as,

$$(0, ...0, \underset{bus}{1}, 0, ..., 0)$$

  - We could represent the word "trolley" as,

$$(0, ..., 0, \underset{bus}{0}, 0, ...0, \underset{trolley}{1}, 0, ...0)$$

Word Embeddings

Basic Idea

- Word embeddings are a form of unsupervised learning, in that they require only the independent variables (or features) X and not labeled data
- Consider first one hot encodings:
  - We could represent the word "bus" as,

$$(0, ...0, \underset{bus}{1}, 0, ..., 0)$$

  - We could represent the word "trolley" as,

$$(0, ..., 0, \underset{bus}{0}, 0, ...0, \underset{trolley}{1}, 0, ...0)$$

  - This representation doesn't know that in many contexts, a bus and a trolley have the same meaning

Word Embeddings

Basic Idea

- Word embeddings are a form of unsupervised learning, in that they require only the independent variables (or features) X and not labeled data
- Consider first one hot encodings:
  - We could represent the word "bus" as,

$$(0, ...0, \underset{bus}{1}, 0, ..., 0)$$

  - We could represent the word "trolley" as,

$$(0, ..., 0, \underset{bus}{0}, 0, ...0, \underset{trolley}{1}, 0, ...0)$$

  - This representation doesn't know that in many contexts, a bus and a trolley have the same meaning
- We could desire a representation that puts these two words close together, allowing us to reduce the number of parameters in our model for y

Word Embeddings

Basic Idea

- Political science example:
  - Inflation was persistently high during Biden's first term

Word Embeddings

Basic Idea

- Political science example:
  - Inflation was persistently high during Biden's first term
  - Inflation was persistently high during the president's first term

Word Embeddings

Basic Idea

- Political science example:
  - Inflation was persistently high during Biden's first term
  - Inflation was persistently high during the president's first term
  - Inflation was consistently high during Biden's first term

Word Embeddings

Basic Idea

- Political science example:
  - Inflation was persistently high during Biden's first term
  - Inflation was persistently high during the president's first term
  - Inflation was consistently high during Biden's first term
  - Price levels were persistently high during Biden's first term

Word Embeddings

Basic Idea

- Political science example:
  - Inflation was persistently high during Biden's first term
  - Inflation was persistently high during the president's first term
  - Inflation was consistently high during Biden's first term
  - Price levels were persistently high during Biden's first term
- Question:
  - How do we know that bus and trolley are similar?

Word Embeddings

Basic Idea

- Political science example:
  - Inflation was persistently high during Biden's first term
  - Inflation was persistently high during the president's first term
  - Inflation was consistently high during Biden's first term
  - Price levels were persistently high during Biden's first term
- Question:
  - How do we know that bus and trolley are similar?
  - How do we know that good and great are similar?

Word Embeddings

Basic Idea

- Political science example:
  - Inflation was persistently high during Biden's first term
  - Inflation was persistently high during the president's first term
  - Inflation was consistently high during Biden's first term
  - Price levels were persistently high during Biden's first term
- Question:
  - How do we know that bus and trolley are similar?
  - How do we know that good and great are similar?
  - How do we know that president and Biden are similar?

Word Embeddings

Basic Idea

- Political science example:
  - Inflation was persistently high during Biden's first term
  - Inflation was persistently high during the president's first term
  - Inflation was consistently high during Biden's first term
  - Price levels were persistently high during Biden's first term
- Question:
  - How do we know that bus and trolley are similar?
  - How do we know that good and great are similar?
  - How do we know that president and Biden are similar?
- Answer: because they tend to show up in similar circumstances

Word Embeddings

Basic Idea

- Some example sentences:
  - I would have taken an Uber, but I took the _____ back home to save money
  - The _____ prospect for reelection doesn't look good if approval among young voters remains as low as it had been

Word Embeddings

Basic Idea

- Some example sentences:
  - I would have taken an Uber, but I took the _____ back home to save money
  - The _____ prospect for reelection doesn't look good if approval among young voters remains as low as it had been
- The word bus fits in the first, but not the second:
  - I would have taken an Uber, but I took the bus back home to save money
  - The bus's prospect for reelection doesn't look good if approval among young voters remains as low as it had been

Word Embeddings

Basic Idea

- Some example sentences:
  - I would have taken an Uber, but I took the _____ back home to save money
  - The _____ prospect for reelection doesn't look good if approval among young voters remains as low as it had been
- The word bus fits in the first, but not the second:
  - I would have taken an Uber, but I took the bus back home to save money
  - The bus's prospect for reelection doesn't look good if approval among young voters remains as low as it had been
- The word trolley fits it the first, but not the second:

Word Embeddings

Basic Idea

- Some example sentences:
  - I would have taken an Uber, but I took the _____ back home to save money
  - The _____ prospect for reelection doesn't look good if approval among young voters remains as low as it had been
- The word bus fits in the first, but not the second:
  - I would have taken an Uber, but I took the bus back home to save money
  - The bus's prospect for reelection doesn't look good if approval among young voters remains as low as it had been
- The word trolley fits it the first, but not the second:
- The word president's fits in the second, but not the first
- The word Biden's fits in the second, but not the first

Word Embeddings

Basic Idea

- Bus and trolley are similar in the sense that they appear in similar contexts, or that they appear around similar groups of words

Word Embeddings

Basic Idea

- Bus and trolley are similar in the sense that they appear in similar contexts, or that they appear around similar groups of words
- Biden and president are similar in the sense that they appear around similar groups of words

Word Embeddings

Basic Idea

- Bus and trolley are similar in the sense that they appear in similar contexts, or that they appear around similar groups of words
- Biden and president are similar in the sense that they appear around similar groups of words
- The word bus is unlikely to be near words like young, voters, approval, etc.

Word Embeddings

Basic Idea

- Bus and trolley are similar in the sense that they appear in similar contexts, or that they appear around similar groups of words
- Biden and president are similar in the sense that they appear around similar groups of words
- The word bus is unlikely to be near words like young, voters, approval, etc.
- The word Biden is unlikely to be around words like Uber, save, money, etc.

Word Embeddings

Basic Idea

- Bus and trolley are similar in the sense that they appear in similar contexts, or that they appear around similar groups of words
- Biden and president are similar in the sense that they appear around similar groups of words
- The word bus is unlikely to be near words like young, voters, approval, etc.
- The word Biden is unlikely to be around words like Uber, save, money, etc.
- How can we capture this?

Word Embeddings

Basic Idea

- Bus and trolley are similar in the sense that they appear in similar contexts, or that they appear around similar groups of words
- Biden and president are similar in the sense that they appear around similar groups of words
- The word bus is unlikely to be near words like young, voters, approval, etc.
- The word Biden is unlikely to be around words like Uber, save, money, etc.
- How can we capture this?
  - Answer: we build a model for the probability of observing a word given a context (or group of surrounding words)

Word Embeddings

Modeling Time Series

- Our data has the form $(w_1, w_2, ..., w_T)$, where the order of the observations is important

Word Embeddings

Modeling Time Series

- Our data has the form $(w_1, w_2, ..., w_T)$, where the order of the observations is important
- The data structure is thus analogous to a time series

Word Embeddings

Modeling Time Series

- Our data has the form $(w_1, w_2, ..., w_T)$, where the order of the observations is important
- The data structure is thus analogous to a time series
- Consider the $AR(1)$ model:

$$y_t | y_{t-1} \sim N(\rho y_{t-1}, \sigma^2)$$

Word Embeddings

Modeling Time Series

- Our data has the form $(w_1, w_2, ..., w_T)$, where the order of the observations is important
- The data structure is thus analogous to a time series
- Consider the $AR(1)$ model:

$$y_t | y_{t-1} \sim N(\rho y_{t-1}, \sigma^2)$$

- In POL 603, we derived,

$$E[y_t] = 0$$

$$Var(y_t) = \frac{\sigma^2}{1-\rho^2}$$

$$Cov(y_t, y_{t-j}) = \frac{\rho^{|j|}\sigma^2}{1-\rho^2}$$

Word Embeddings

Modeling Time Series

- This allowed us to determine that,

$$y \sim N(0, \Omega)$$

where $\Omega_{t,s} = \frac{\rho^{|t-s|}\sigma^2}{1-\rho^2}$

Word Embeddings

Modeling Time Series

- This allowed us to determine that,

$$y \sim N(0, \Omega)$$

where $\Omega_{t,s} = \frac{\rho^{|t-s|}\sigma^2}{1-\rho^2}$

- We could then estimate (ρ_0, σ_0^2) using the MLE,

$$(\hat{\rho}_{MLE}, \hat{\sigma}^2_{MLE}) = \underset{(\rho,\sigma^2)}{\arg\max}\, log f(y; \rho, \sigma^2)$$

$$= \underset{(\rho,\sigma^2)}{\arg\max}\, log \left(\frac{1}{(2\pi)^{T/2} det(\Omega(\rho, \sigma^2))^{1/2}} e^{-\frac{1}{2} y' \Omega(\rho,\sigma^2)^{-1} y} \right)$$

Word Embeddings

Modeling Time Series

- An alternative derivation relies on conditional distributions:

$$f(y;\rho,\sigma^2) = f(y_1;\rho,\sigma^2)f(y_2|y_1;\rho,\sigma^2)f(y_3|y_1,y_2;\rho,\sigma^2)$$

$$* \cdots * f(y_T|y_1,...,y_{T-1};\rho,\sigma^2)$$

Word Embeddings

Modeling Time Series

- An alternative derivation relies on conditional distributions:

$$f(y;\rho,\sigma^2) = f(y_1;\rho,\sigma^2)f(y_2|y_1;\rho,\sigma^2)f(y_3|y_1,y_2;\rho,\sigma^2)$$

$$* \cdots * f(y_T|y_1,...,y_{T-1};\rho,\sigma^2)$$

$$= f(y_1;\rho,\sigma^2)f(y_2|y_1;\rho,\sigma^2)f(y_3|y_2;\rho,\sigma^2)*\cdots*f(y_T|y_{T-1};\rho,\sigma^2)$$

Word Embeddings

Modeling Time Series

- This allows us to define the MLE as,

$$(\hat{\rho}_{MLE}, \hat{\sigma}^2_{MLE}) = \underset{(\rho,\sigma^2)}{\arg\max}\, log f(y_1; \rho, \sigma^2) + \sum_{t=2}^{T} log f(y_t | y_{t-1}; \rho, \sigma^2)$$

Word Embeddings

Modeling Time Series

- This allows us to define the MLE as,

$$(\hat{\rho}_{MLE}, \hat{\sigma}^2_{MLE}) = \arg\max_{(\rho,\sigma^2)} log f(y_1; \rho, \sigma^2) + \sum_{t=2}^{T} log f(y_t | y_{t-1}; \rho, \sigma^2)$$

$$= \arg\max_{(\rho,\sigma^2)} log \frac{1}{\sigma\sqrt{2\pi(1-\rho^2)}} e^{-\frac{1}{(1-\rho^2)\sigma^2} y_1^2} + \sum_{t=2}^{T} log \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2\sigma^2}(y_t - \rho y_{t-1})^2}$$

$$= \arg\max_{(\rho,\sigma^2)} -log(\sigma\sqrt{1-\rho^2}) - \frac{y_1^2}{(1-\rho^2)\sigma^2} - (T-1)log(\sigma) - \sum_{t=2}^{T} \frac{(y_t - \rho y_{t-1})^2}{2\sigma^2}$$

Word Embeddings

Modeling Time Series

- An alternative estimator is the Cochrane-Orcutt estimator, which drops the y_1 term from the log-likelihood:

$$(\hat{\rho}_{CU}, \hat{\sigma}^2_{CU}) = \underset{(\rho,\sigma^2)}{\arg\max} -(T-1)log(\sigma) - \sum_{t=2}^{T} \frac{(y_t - \rho y_{t-1})^2}{2\sigma^2}$$

Word Embeddings

Modeling Time Series

- An alternative estimator is the Cochrane-Orcutt estimator, which drops the y_1 term from the log-likelihood:

$$(\hat{\rho}_{CU}, \hat{\sigma}^2_{CU}) = \underset{(\rho,\sigma^2)}{\arg\max} -(T-1)log(\sigma) - \sum_{t=2}^{T} \frac{(y_t - \rho y_{t-1})^2}{2\sigma^2}$$

- In large samples, the Cochrane-Orcutt estimator is equivalent to the MLE (hence, it is consistent, asymptotically normal, and efficient)

Word Embeddings

Modeling Time Series

- An alternative estimator is the Cochrane-Orcutt estimator, which drops the y_1 term from the log-likelihood:

$$(\hat{\rho}_{CU}, \hat{\sigma}^2_{CU}) = \underset{(\rho,\sigma^2)}{\arg\max} -(T-1)log(\sigma) - \sum_{t=2}^{T} \frac{(y_t - \rho y_{t-1})^2}{2\sigma^2}$$

- In large samples, the Cochrane-Orcutt estimator is equivalent to the MLE (hence, it is consistent, asymptotically normal, and efficient)
- Historically, the Cochrane-Orcutt was used because it was easier to compute

Word Embeddings

Modeling Time Series

- An alternative estimator is the Cochrane-Orcutt estimator, which drops the y_1 term from the log-likelihood:

$$(\hat{\rho}_{CU}, \hat{\sigma}^2_{CU}) = \underset{(\rho,\sigma^2)}{\arg\max} -(T-1)log(\sigma) - \sum_{t=2}^{T} \frac{(y_t - \rho y_{t-1})^2}{2\sigma^2}$$

- In large samples, the Cochrane-Orcutt estimator is equivalent to the MLE (hence, it is consistent, asymptotically normal, and efficient)
- Historically, the Cochrane-Orcutt was used because it was easier to compute
- It also made derivations easier

Word Embeddings

Modeling Time Series

- An alternative estimator is the Cochrane-Orcutt estimator, which drops the y_1 term from the log-likelihood:

$$(\hat{\rho}_{CU}, \hat{\sigma}^2_{CU}) = \underset{(\rho,\sigma^2)}{\arg\max} -(T-1)log(\sigma) - \sum_{t=2}^{T} \frac{(y_t - \rho y_{t-1})^2}{2\sigma^2}$$

- In large samples, the Cochrane-Orcutt estimator is equivalent to the MLE (hence, it is consistent, asymptotically normal, and efficient)
- Historically, the Cochrane-Orcutt was used because it was easier to compute
- It also made derivations easier
- In time series, increasing computational power means Cochrane-Orcutt is rarely used

Word Embeddings

Modeling Time Series

- An alternative estimator is the Cochrane-Orcutt estimator, which drops the y_1 term from the log-likelihood:

$$(\hat{\rho}_{CU}, \hat{\sigma}^2_{CU}) = \underset{(\rho,\sigma^2)}{\arg\max} -(T-1)log(\sigma) - \sum_{t=2}^{T} \frac{(y_t - \rho y_{t-1})^2}{2\sigma^2}$$

- In large samples, the Cochrane-Orcutt estimator is equivalent to the MLE (hence, it is consistent, asymptotically normal, and efficient)
- Historically, the Cochrane-Orcutt was used because it was easier to compute
- It also made derivations easier
- In time series, increasing computational power means Cochrane-Orcutt is rarely used
- In text analysis, computational power is much more of a concern, so a similar principle in useful

Word Embeddings

Modeling Sentence Structure

- Following the discussion above, we build a model for $\Pr(w_2, w_3, ..., w_T | w_1)$

Word Embeddings

Modeling Sentence Structure

- Following the discussion above, we build a model for $\Pr(w_2, w_3, ..., w_T|w_1)$
- As a first cut, we assume that $\Pr(w_t|w_{t-1}, w_{t-2}, ..., w_1) = \Pr(w_t|w_{t-1})$ (i.e. we assume that conditional on the previous word, the current word is independent of words that are two units away)

Word Embeddings

Modeling Sentence Structure

- Analogizing to ideal point models, we let α_w be the latent location of word w

Word Embeddings

Modeling Sentence Structure

- Analogizing to ideal point models, we let α_w be the latent location of word w
- We can specify,

$$\Pr(w_t|w_{t-1}) = \frac{e^{a_{w_t} - (\alpha_{w_t} - \alpha_{w_{t-1}})' W (\alpha_{w_t} - \alpha_{w_{t-1}})}}{\sum_{w=1}^{W} e^{a_w - (\alpha_w - \alpha_{w_{t-1}})' W (\alpha_w - \alpha_{w_{t-1}})}}$$

where W is a positive definite weighting matrix, $(\alpha_w - \alpha_{w_{t-1}})' W (\alpha_w - \alpha_{w_{t-1}})$ is the distance between word w and word w_{t-1}, and a_w controls the overall frequency of the word

Word Embeddings

Modeling Sentence Structure

- Analogizing to ideal point models, we let α_w be the latent location of word w
- We can specify,

$$\Pr(w_t|w_{t-1}) = \frac{e^{a_{w_t} - (\alpha_{w_t} - \alpha_{w_{t-1}})' W (\alpha_{w_t} - \alpha_{w_{t-1}})}}{\sum_{w=1}^{W} e^{a_w - (\alpha_w - \alpha_{w_{t-1}})' W (\alpha_w - \alpha_{w_{t-1}})}}$$

  where W is a positive definite weighting matrix, $(\alpha_w - \alpha_{w_{t-1}})' W (\alpha_w - \alpha_{w_{t-1}})$ is the distance between word w and word w_{t-1}, and a_w controls the overall frequency of the word

- This model is very similar to the model of Barbera (2015)

Word Embeddings

Modeling Sentence Structure

- From our discussion of ideal point estimation, if we are only interested in prediction (and not interpretation), it is computationally useful to simplify the model by introducing a reparameterization:

$$a_{w_t} - (\alpha_{w_t} - \alpha_{w_{t-1}})' W (\alpha_{w_t} - \alpha_{w_{t-1}})$$

$$= a_{w_t} - \alpha'_{w_t} W \alpha_{w_t} - \alpha'_{w_{t-1}} W \alpha_{w_{t-1}} + 2\alpha'_{w_{t-1}} W \alpha_{w_t}$$

$$= (a_t - \alpha'_{w_t} W \alpha_{w_t}) 1 - 1(\alpha'_{w_{t-1}} W \alpha_{w_{t-1}}) + (2\alpha'_{w_{t-1}} W) \alpha_{w_t} = u'_{w_t} v_{w_{t-1}}$$

where

$$u_{w_t} = (a_{w_t} - \alpha'_{w_t} W \alpha_{w_t}, 1, 2\alpha'_{w_{t-1}} W)$$

$$v_{w_{t-1}} = (1, \alpha'_{w_{t-1}} W \alpha_{w_{t-1}}, \alpha_{w_t})$$

Word Embeddings

Modeling Sentence Structure

- In the reparameterized model, we have,

$$\Pr(w_t|w_{t-1}) = \frac{e^{u'_{w_t} v_{w_{t-1}}}}{\sum_{w=1}^{W} e^{u'_w v_{w_{t-1}}}}$$

Word Embeddings

Modeling Sentence Structure

- In the reparameterized model, we have,

$$\Pr(w_t|w_{t-1}) = \frac{e^{u'_{w_t} v_{w_{t-1}}}}{\sum_{w=1}^{W} e^{u'_w v_{w_{t-1}}}}$$

- The number of parameters has increased:
  - In the original model, $dim(\alpha) = VW$, $dim(W) = V(V+1)/2$, and $dim(a) = W$, so that $dim(\alpha, a, W) = (V+1)(W+V/2)$

Word Embeddings

Modeling Sentence Structure

- In the reparameterized model, we have,

$$\Pr(w_t|w_{t-1}) = \frac{e^{u'_{w_t} v_{w_{t-1}}}}{\sum_{w=1}^{W} e^{u'_w v_{w_{t-1}}}}$$

- The number of parameters has increased:
  - In the original model, $dim(\alpha) = VW$, $dim(W) = V(V+1)/2$, and $dim(a) = W$, so that $dim(\alpha, a, W) = (V+1)(W+V/2)$
  - In the reparameterized model, $dim((u, v)) = 2VW$

Word Embeddings

Modeling Sentence Structure

- In the reparameterized model, we have,

$$\Pr(w_t|w_{t-1}) = \frac{e^{u'_{w_t} v_{w_{t-1}}}}{\sum_{w=1}^{W} e^{u'_w v_{w_{t-1}}}}$$

- The number of parameters has increased:
  - In the original model, $dim(\alpha) = VW$, $dim(W) = V(V+1)/2$, and $dim(a) = W$, so that $dim(\alpha, a, W) = (V+1)(W+V/2)$
  - In the reparameterized model, $dim((u, v)) = 2VW$
- However, the derivatives of the new model are much simpler, leading to a model that is much easier to fit

Word Embeddings

Modeling Sentence Structure

- As a next step, we may want to allow the current word to depend on more than one previous word:

$$\Pr(w_t|w_{t-1}, w_{t-2}, ..., w_1) = \Pr(w_t|w_{t-1}, w_{t-2}, ..., w_{t-c})$$

Word Embeddings

Modeling Sentence Structure

- As a next step, we may want to allow the current word to depend on more than one previous word:

$$\Pr(w_t|w_{t-1}, w_{t-2}, ..., w_1) = \Pr(w_t|w_{t-1}, w_{t-2}, ..., w_{t-c})$$

- We could specify,

$$\Pr(w_t|w_{t-1}, ..., w_{t-c}) = \frac{e^{\sum_{j=1}^{c} \delta_j u'_{w_t} v_{w_{t-j}}}}{\sum_{w=1}^{W} e^{\sum_{j=1}^{c} \delta_j u'_w v_{w_{t-j}}}}$$

$$= \frac{e^{u'_{w_t} \sum_{j=1}^{c} \delta_j v_{w_{t-j}}}}{\sum_{w=1}^{W} e^{u'_w \sum_{j=1}^{c} \delta_j v_{w_{t-j}}}} = \frac{e^{u'_{w_t} \bar{v_t}}}{\sum_{w=1}^{W} e^{u'_w \bar{v_t}}}$$

where $\bar{v_t} = \sum_{j=1}^{c} \delta_j v_{w_{t-j}}$

Word Embeddings

Modeling Sentence Structure

- As a next step, we may want to allow the current word to depend on more than one previous word:

$$\Pr(w_t|w_{t-1}, w_{t-2}, ..., w_1) = \Pr(w_t|w_{t-1}, w_{t-2}, ..., w_{t-c})$$

- We could specify,

$$\Pr(w_t|w_{t-1}, ..., w_{t-c}) = \frac{e^{\sum_{j=1}^{c} \delta_j u'_{w_t} v_{w_{t-j}}}}{\sum_{w=1}^{W} e^{\sum_{j=1}^{c} \delta_j u'_w v_{w_{t-j}}}}$$

$$= \frac{e^{u'_{w_t} \sum_{j=1}^{c} \delta_j v_{w_{t-j}}}}{\sum_{w=1}^{W} e^{u'_w \sum_{j=1}^{c} \delta_j v_{w_{t-j}}}} = \frac{e^{u'_{w_t} \bar{v_t}}}{\sum_{w=1}^{W} e^{u'_w \bar{v_t}}}$$

  where $\bar{v_t} = \sum_{j=1}^{c} \delta_j v_{w_{t-j}}$

- Here, δ_j are weights, with $\delta_j > 0$ and $\sum_{j=1}^{c} \delta_j = 1$, and typically $\delta_j = \frac{1}{c}$

Word Embeddings

Modeling Sentence Structure

- This approach is called continuous bag of words, or CBOW, because it forms a prediction by averaging word embeddings
- To complete the model, we specify,

$$(\hat{u}, \hat{v}) = \arg\max_{(u,v)} \Pr(w_{c+1}, ..., w_T | w_1, ..., w_c)$$

$$= \arg\max_{(u,v)} \sum_{t=c+1}^{T} log \frac{e^{u'_{w_t} \bar{v}_t}}{\sum_{w=1}^{W} e^{u'_w \bar{v}_t}}$$

Word Embeddings

Modeling Sentence Structure

- Under the skip-gram approach, we have,

$$(\hat{u}, \hat{v}) = \arg\max_{(u,v)} \sum_{t=c+1}^{T} \sum_{j=1}^{c} log \Pr(w_t | w_{t-j})$$

$$= \arg\max_{(u,v)} \sum_{t=c+1}^{T} \sum_{j=1}^{c} log \frac{e^{u'_{w_t} v_{w_{t-j}}}}{\sum_{w=1}^{W} e^{u'_w v_{w_{t-j}}}}$$

Word Embeddings

Modeling Sentence Structure

- Under the skip-gram approach, we have,

$$(\hat{u}, \hat{v}) = \arg\max_{(u,v)} \sum_{t=c+1}^{T} \sum_{j=1}^{c} log \Pr(w_t | w_{t-j})$$

$$= \arg\max_{(u,v)} \sum_{t=c+1}^{T} \sum_{j=1}^{c} log \frac{e^{u'_{w_t} v_{w_{t-j}}}}{\sum_{w=1}^{W} e^{u'_w v_{w_{t-j}}}}$$

- This approach deviates somewhat from forming a probability model for how words are generated

Word Embeddings

Word2Vec Embeddings

- word2vec makes some further small changes to the models above:

$$(\hat{u}, \hat{v}) = \arg\max_{(u,v)} \sum_{t=c+1}^{T-c} log \frac{e^{u'_{w_t} \bar{v}_t}}{\sum_{w=1}^{W} e^{u'_w \bar{v}_t}}$$

where $\bar{v}_t = \sum_{-c \leq j \leq c, j \neq 0} v_{w_{t-j}}$

Word Embeddings

Word2Vec Embeddings

- word2vec makes some further small changes to the models above:

$$(\hat{u}, \hat{v}) = \arg\max_{(u,v)} \sum_{t=c+1}^{T-c} log \frac{e^{u'_{w_t} \bar{v}_t}}{\sum_{w=1}^{W} e^{u'_w \bar{v}_t}}$$

where $\bar{v}_t = \sum_{-c \leq j \leq c, j \neq 0} v_{w_{t-j}}$

$$(\hat{u}, \hat{v}) = \arg\max_{(u,v)} \sum_{t=c+1}^{T-c} \sum_{-c \leq j \leq c, j \neq 0} log \frac{e^{u'_{w_t} v_{w_{t-j}}}}{\sum_{w=1}^{W} e^{u'_w v_{w_{t-j}}}}$$

Word Embeddings

Word2Vec Embeddings

- word2vec makes some further small changes to the models above:

$$(\hat{u}, \hat{v}) = \arg\max_{(u,v)} \sum_{t=c+1}^{T-c} log \frac{e^{u'_{w_t} \bar{v}_t}}{\sum_{w=1}^{W} e^{u'_w \bar{v}_t}}$$

where $\bar{v}_t = \sum_{-c \leq j \leq c, j \neq 0} v_{w_{t-j}}$

$$(\hat{u}, \hat{v}) = \arg\max_{(u,v)} \sum_{t=c+1}^{T-c} \sum_{-c \leq j \leq c, j \neq 0} log \frac{e^{u'_{w_t} v_{w_{t-j}}}}{\sum_{w=1}^{W} e^{u'_w v_{w_{t-j}}}}$$

- Note that in both cases, the computational complexity of computing the objective function is $O(WVTc)$

Word Embeddings

Word2Vec Embeddings

- To avoid the computational complexity, instead of computing $\sum_{w=1}^{W} e^{u'_w v_{w_{t-j}}}$, we compute the same expression on a sample of words

Word Embeddings

Word2Vec Embeddings

- To avoid the computational complexity, instead of computing $\sum_{w=1}^{W} e^{u'_w v_{w_{t-j}}}$, we compute the same expression on a sample of words
- McFadden (1978) shows that because the multinomial logit model satisfies the IIA property, we can randomly sample choices and maintain consistency of the estimator

Word Embeddings

Word2Vec Embeddings

- To avoid the computational complexity, instead of computing $\sum_{w=1}^{W} e^{u'_w v_{w_{t-j}}}$, we compute the same expression on a sample of words
- McFadden (1978) shows that because the multinomial logit model satisfies the IIA property, we can randomly sample choices and maintain consistency of the estimator
  - Specifically, for each t, sample W^* words without replacement, and denote these words with the set $\mathcal{W}_t$

Word Embeddings

Word2Vec Embeddings

- To avoid the computational complexity, instead of computing $\sum_{w=1}^{W} e^{u'_w v_{w_{t-j}}}$, we compute the same expression on a sample of words
- McFadden (1978) shows that because the multinomial logit model satisfies the IIA property, we can randomly sample choices and maintain consistency of the estimator
  - Specifically, for each t, sample W^* words without replacement, and denote these words with the set $\mathcal{W}_t$
- We have,

$$(\hat{u}, \hat{v}) = \arg\max_{(u,v)} \sum_{t=c+1}^{T-c} \sum_{-c \leq j \leq c, j \neq 0} log \frac{e^{u'_{w_t} v_{w_{t-j}}}}{\sum_{w \in \mathcal{W}_t} e^{u'_w v_{w_{t-j}}}}$$

Word Embeddings

Word2Vec Embeddings

- To avoid the computational complexity, instead of computing $\sum_{w=1}^{W} e^{u'_w v_{w_{t-j}}}$, we compute the same expression on a sample of words
- McFadden (1978) shows that because the multinomial logit model satisfies the IIA property, we can randomly sample choices and maintain consistency of the estimator
  - Specifically, for each t, sample W^* words without replacement, and denote these words with the set $\mathcal{W}_t$
- We have,

$$(\hat{u}, \hat{v}) = \arg\max_{(u,v)} \sum_{t=c+1}^{T-c} \sum_{-c \leq j \leq c, j \neq 0} log \frac{e^{u'_{w_t} v_{w_{t-j}}}}{\sum_{w \in \mathcal{W}_t} e^{u'_w v_{w_{t-j}}}}$$

- Mikolov et al. (2013) recommend sampling word w with probability $f_w^{3/4} / \sum_{w'=1}^{W} f_{w'}^{3/4}$

Word Embeddings

Word2Vec Embeddings

- Mikolov et al. (2013) also recommend dropping words with probability $1 - \sqrt{\frac{\omega}{f_w}}$, where $\omega = 10^{-5}$.

Word Embeddings

Word2Vec Embeddings

- Mikolov et al. (2013) also recommend dropping words with probability $1 - \sqrt{\frac{\omega}{f_w}}$, where $\omega = 10^{-5}$.
- Fitting the model is helped by computing the analytical gradient

Word Embeddings

Word2Vec Embeddings

- Mikolov et al. (2013) also recommend dropping words with probability $1 - \sqrt{\frac{\omega}{f_w}}$, where $\omega = 10^{-5}$.
- Fitting the model is helped by computing the analytical gradient
  - As usual, the computational complexity of computing the gradient is the same as computing the objective function (Griewank and Walther, 2008)

Word Embeddings

Word2Vec Embeddings

- Mikolov et al. (2013) also recommend dropping words with probability $1 - \sqrt{\frac{\omega}{f_w}}$, where $\omega = 10^{-5}$.
- Fitting the model is helped by computing the analytical gradient
  - As usual, the computational complexity of computing the gradient is the same as computing the objective function (Griewank and Walther, 2008)
- In principle, we could compute u and v via full optimization

Word Embeddings

Word2Vec Embeddings

- Mikolov et al. (2013) also recommend dropping words with probability $1 - \sqrt{\frac{\omega}{f_w}}$, where $\omega = 10^{-5}$.
- Fitting the model is helped by computing the analytical gradient
  - As usual, the computational complexity of computing the gradient is the same as computing the objective function (Griewank and Walther, 2008)
- In principle, we could compute u and v via full optimization
- In practice, methods similar to the ones we covered for deep learning are used—stochastic gradient descent is employed, a fixed number of epochs are used, etc.

Word Embeddings

Word2Vec Embeddings

- Mikolov et al. (2013) also recommend dropping words with probability $1 - \sqrt{\frac{\omega}{f_w}}$, where $\omega = 10^{-5}$.
- Fitting the model is helped by computing the analytical gradient
  - As usual, the computational complexity of computing the gradient is the same as computing the objective function (Griewank and Walther, 2008)
- In principle, we could compute u and v via full optimization
- In practice, methods similar to the ones we covered for deep learning are used—stochastic gradient descent is employed, a fixed number of epochs are used, etc.
- The resulting embedding is given by u or $\frac{1}{2}(u + v)$

Word Embeddings

Cosine Distance

- The Cauchy-Schwarz inequality tells us that $|u'v| \leq ||u||_2||v||_2$ where equality occurs only if there exists an α such that $v = \alpha u$

Word Embeddings

Cosine Distance

- The Cauchy-Schwarz inequality tells us that $|u'v| \leq ||u||_2||v||_2$ where equality occurs only if there exists an α such that $v = \alpha u$
- Cosine similarity of the vector u and v is define by,

$$CosSim = \frac{u'v}{||u||_2||v||_2}$$

Word Embeddings

Cosine Distance

- The Cauchy-Schwarz inequality tells us that $|u'v| \leq ||u||_2||v||_2$ where equality occurs only if there exists an α such that $v = \alpha u$
- Cosine similarity of the vector u and v is define by,

$$CosSim = \frac{u'v}{||u||_2||v||_2}$$

  - Note that $|u'v| \leq ||u||_2||v||_2$ implies that $u'v \leq ||u||_2||v||_2$, which implies that $CosSim \leq 1$

Word Embeddings

Cosine Distance

- The Cauchy-Schwarz inequality tells us that $|u'v| \leq ||u||_2||v||_2$ where equality occurs only if there exists an α such that $v = \alpha u$
- Cosine similarity of the vector u and v is define by,

$$CosSim = \frac{u'v}{||u||_2||v||_2}$$

  - Note that $|u'v| \leq ||u||_2||v||_2$ implies that $u'v \leq ||u||_2||v||_2$, which implies that $CosSim \leq 1$
  - Note further that if there exists an α such that $v = \alpha u$, $CosSim = 1$

Word Embeddings

Cosine Distance

- $CosSim$ measures whether two vectors point is the same direction

Word Embeddings

Cosine Distance

- $CosSim$ measures whether two vectors point is the same direction
  - If the vectors being compared have the same size, or if there elements sum to 1, then no vector will satisfy $v = \alpha u$, so it will be a traditional measure of closeness

Word Embeddings

Cosine Distance

- $CosSim$ measures whether two vectors point is the same direction
  - If the vectors being compared have the same size, or if there elements sum to 1, then no vector will satisfy $v = \alpha u$, so it will be a traditional measure of closeness
  - For example, discrete probabilities sum to 1, or word frequencies sum to 1

Word Embeddings

Cosine Distance

- $CosSim$ measures whether two vectors point is the same direction
  - If the vectors being compared have the same size, or if there elements sum to 1, then no vector will satisfy $v = \alpha u$, so it will be a traditional measure of closeness
  - For example, discrete probabilities sum to 1, or word frequencies sum to 1
  - In principle, we could require u to take values on the unit circle and our word embedding could be interpreted with cosine similarity as a measure of closeness

Word Embeddings

Cosine Distance

- $CosSim$ measures whether two vectors point is the same direction
  - If the vectors being compared have the same size, or if there elements sum to 1, then no vector will satisfy $v = \alpha u$, so it will be a traditional measure of closeness
  - For example, discrete probabilities sum to 1, or word frequencies sum to 1
  - In principle, we could require u to take values on the unit circle and our word embedding could be interpreted with cosine similarity as a measure of closeness
  - Not enforcing this constraint could be viewed as a way of getting at a similar idea while not imposing the additional computational cost of maintaining the constraint

Word Embeddings

Cosine Distance

- word2vec assumes two words are likely to appear near each other if one word's u is near another word's v:

$$(\hat{u}, \hat{v}) = \arg\max_{(u,v)} \sum_{t=c+1}^{T-c} \sum_{-c \leq j \leq c, j \neq 0} log \frac{e^{u'_{w_t} v_{w_{t-j}}}}{\sum_{w \in \mathcal{W}_t} e^{u'_w v_{w_{t-j}}}}$$

Word Embeddings

GloVe Embeddings

- Consider the co-occurrence matrix, Z, where Z_{ij} counts the number of times word j appears within c words of word i across all documents

Word Embeddings

GloVe Embeddings

- Consider the co-occurrence matrix, Z, where Z_{ij} counts the number of times word j appears within c words of word i across all documents
- Z is a W by W matrix, where W is the size of the vocabulary

Word Embeddings

GloVe Embeddings

- Consider the co-occurrence matrix, Z, where Z_{ij} counts the number of times word j appears within c words of word i across all documents
- Z is a W by W matrix, where W is the size of the vocabulary
- We can reduce the dimensionality of Z using a singular value decomposition, $Z = U\Psi V'$

Word Embeddings

GloVe Embeddings

- Consider the co-occurrence matrix, Z, where Z_{ij} counts the number of times word j appears within c words of word i across all documents
- Z is a W by W matrix, where W is the size of the vocabulary
- We can reduce the dimensionality of Z using a singular value decomposition, $Z = U\Psi V'$
- Here, the first D columns of U can serve as the word embeddings

Word Embeddings

GloVe Embeddings

- Consider the co-occurrence matrix, Z, where Z_{ij} counts the number of times word j appears within c words of word i across all documents
- Z is a W by W matrix, where W is the size of the vocabulary
- We can reduce the dimensionality of Z using a singular value decomposition, $Z = U\Psi V'$
- Here, the first D columns of U can serve as the word embeddings
- The matrix Z will typically be sparse—we can let S denote the number of nonzero entries in Z

Word Embeddings

GloVe Embeddings

- Consider the co-occurrence matrix, Z, where Z_{ij} counts the number of times word j appears within c words of word i across all documents
- Z is a W by W matrix, where W is the size of the vocabulary
- We can reduce the dimensionality of Z using a singular value decomposition, $Z = U\Psi V'$
- Here, the first D columns of U can serve as the word embeddings
- The matrix Z will typically be sparse—we can let S denote the number of nonzero entries in Z
- The cost of computing the SVD is $O(S)$ per iteration, per factor

Word Embeddings

GloVe Embeddings

- Consider the co-occurrence matrix, Z, where Z_{ij} counts the number of times word j appears within c words of word i across all documents
- Z is a W by W matrix, where W is the size of the vocabulary
- We can reduce the dimensionality of Z using a singular value decomposition, $Z = U\Psi V'$
- Here, the first D columns of U can serve as the word embeddings
- The matrix Z will typically be sparse—we can let S denote the number of nonzero entries in Z
- The cost of computing the SVD is $O(S)$ per iteration, per factor
- If D factors are desired, the cost is $O(SD)$ per iteration

Word Embeddings

GloVe Embeddings

- Why SVD rather than word2vec?
  - Word co-occurrence matrix can be computed once rather than multiple times during training

Word Embeddings

GloVe Embeddings

- Why SVD rather than word2vec?
  - Word co-occurrence matrix can be computed once rather than multiple times during training
- Why not SVD?
  - S can still be quite big
  - Co-occurrence patterns can be dominated by low information words

Word Embeddings

GloVe Embeddings

- While SVD can work, Pennington, Socher and Manning (2014) suggests some potential improvements to the basic framework

Word Embeddings

GloVe Embeddings

- While SVD can work, Pennington, Socher and Manning (2014) suggests some potential improvements to the basic framework
- The GloVe (global vector) model considers the objective function:

$$J(a, b, u, v) = \sum_{i=1}^{W} \sum_{j=1}^{W} f(Z_{ij})(a_i + b_j + u_i' v_j - log Z_{ij})^2$$

Word Embeddings

GloVe Embeddings

- While SVD can work, Pennington, Socher and Manning (2014) suggests some potential improvements to the basic framework
- The GloVe (global vector) model considers the objective function:

$$J(a, b, u, v) = \sum_{i=1}^{W} \sum_{j=1}^{W} f(Z_{ij})(a_i + b_j + u_i' v_j - log Z_{ij})^2$$

- The terms a_i and b_j handle the overall frequencies of words

Word Embeddings

GloVe Embeddings

- While SVD can work, Pennington, Socher and Manning (2014) suggests some potential improvements to the basic framework
- The GloVe (global vector) model considers the objective function:

$$J(a, b, u, v) = \sum_{i=1}^{W} \sum_{j=1}^{W} f(Z_{ij})(a_i + b_j + u_i' v_j - logZ_{ij})^2$$

- The terms a_i and b_j handle the overall frequencies of words
- $logZ_{ij}$ attempts to reduce the impact of common co-occurrences (such as "the" and "a", which don't provide useful information)

Word Embeddings

GloVe Embeddings

- The weights,

$$f(z) = \begin{cases} (z/z_{max})^{\alpha} & z < z_{max} \\ 1 & z \geq z_{max} \end{cases}$$

where $z_{max} = 100$ and $\alpha = \frac{3}{4}$ also attempt to reduce the impact of common co-occurrences

Word Embeddings

GloVe Embeddings

- The weights,

$$f(z) = \begin{cases} (z/z_{max})^{\alpha} & z < z_{max} \\ 1 & z \geq z_{max} \end{cases}$$

  where $z_{max} = 100$ and $\alpha = \frac{3}{4}$ also attempt to reduce the impact of common co-occurrences
- The resulting embedding is given by u or $\frac{1}{2}(u + v)$

Word Embeddings

GloVe Embeddings

- The weights,

$$f(z) = \begin{cases} (z/z_{max})^{\alpha} & z < z_{max} \\ 1 & z \geq z_{max} \end{cases}$$

  where $z_{max} = 100$ and $\alpha = \frac{3}{4}$ also attempt to reduce the impact of common co-occurrences
- The resulting embedding is given by u or $\frac{1}{2}(u + v)$
- The model is trained in ways that are similar to neural networks

Word Embeddings

GloVe Embeddings

- The weights,

$$f(z) = \begin{cases} (z/z_{max})^{\alpha} & z < z_{max} \\ 1 & z \geq z_{max} \end{cases}$$

  where $z_{max} = 100$ and $\alpha = \frac{3}{4}$ also attempt to reduce the impact of common co-occurrences
- The resulting embedding is given by u or $\frac{1}{2}(u + v)$
- The model is trained in ways that are similar to neural networks
- Unlike word2vec, one does not need to iterate through the corpus multiple times in training

Word Embeddings

- Why embeddings may help?
  - Word embeddings work off of text alone and there is a very large amount of text use as data

Word Embeddings

- Why embeddings may help?
  - Word embeddings work off of text alone and there is a very large amount of text use as data
  - Supervised learning applied to text requires labeled text, which can be costly to obtain, so there is less of it

Word Embeddings

- Why embeddings may help?
  - Word embeddings work off of text alone and there is a very large amount of text use as data
  - Supervised learning applied to text requires labeled text, which can be costly to obtain, so there is less of it
  - Supervised learning applied to text technically has an embedding—the map between words and the first layer of neurons—which is estimated from the labeled data

Word Embeddings

- Why embeddings may help?
  - Word embeddings work off of text alone and there is a very large amount of text use as data
  - Supervised learning applied to text requires labeled text, which can be costly to obtain, so there is less of it
  - Supervised learning applied to text technically has an embedding—the map between words and the first layer of neurons—which is estimated from the labeled data
  - If we use (pre-trained) embeddings, we can reduce the number of parameters we need to estimate using the labeled data

Word Embeddings

- Why embeddings may help?
  - Word embeddings work off of text alone and there is a very large amount of text use as data
  - Supervised learning applied to text requires labeled text, which can be costly to obtain, so there is less of it
  - Supervised learning applied to text technically has an embedding—the map between words and the first layer of neurons—which is estimated from the labeled data
  - If we use (pre-trained) embeddings, we can reduce the number of parameters we need to estimate using the labeled data
  - Using (pre-trained) embeddings, we are also relying on work that someone else has done, reducing the computational burden in our application

Deep Learning for Text Analysis

- Does using embeddings help with over-fitting?

Deep Learning for Text Analysis

- Does using embeddings help with over-fitting?
  - Using BOW, we have a layer with WK parameters

Deep Learning for Text Analysis

- Does using embeddings help with over-fitting?
  - Using BOW, we have a layer with WK parameters
  - Let's consider using a $B = 100$ dimensional GloVe embedding that was trained on 6 billion tokens

Deep Learning for Text Analysis

- Does using embeddings help with over-fitting?
  - Using BOW, we have a layer with WK parameters
  - Let's consider using a $B = 100$ dimensional GloVe embedding that was trained on 6 billion tokens
  - For each word in the document, we compute an 100-dimensional vector—we then average these vectors over words in the document, using the averages as features

Deep Learning for Text Analysis

- Does using embeddings help with over-fitting?
  - Using BOW, we have a layer with WK parameters
  - Let's consider using a $B = 100$ dimensional GloVe embedding that was trained on 6 billion tokens
  - For each word in the document, we compute an 100-dimensional vector—we then average these vectors over words in the document, using the averages as features
  - The approach is referred to as CBOW—using CBOW, we replace a layer with $(W + 1)K$ parameters with one with $(B + 1)K$ parameters

Deep Learning for Text Analysis

- Does using embeddings help with over-fitting?
  - Using BOW, we have a layer with WK parameters
  - Let's consider using a $B = 100$ dimensional GloVe embedding that was trained on 6 billion tokens
  - For each word in the document, we compute an 100-dimensional vector—we then average these vectors over words in the document, using the averages as features
  - The approach is referred to as CBOW—using CBOW, we replace a layer with $(W + 1)K$ parameters with one with $(B + 1)K$ parameters
  - If W is much less than B (as it typically is), this reduces the number of parameters considerably

Deep Learning for Text Analysis

- Does using embeddings help with over-fitting?
  - Using BOW, we have a layer with WK parameters
  - Let's consider using a $B = 100$ dimensional GloVe embedding that was trained on 6 billion tokens
  - For each word in the document, we compute an 100-dimensional vector—we then average these vectors over words in the document, using the averages as features
  - The approach is referred to as CBOW—using CBOW, we replace a layer with $(W + 1)K$ parameters with one with $(B + 1)K$ parameters
  - If W is much less than B (as it typically is), this reduces the number of parameters considerably
  - In Application 1, we have $W = 1,534$, so we reduce the number of parameters by $(W - B)K = 286,800$

Deep Learning for Text Analysis

- Does using embeddings help with over-fitting?
  - Using BOW, we have a layer with WK parameters
  - Let's consider using a $B = 100$ dimensional GloVe embedding that was trained on 6 billion tokens
  - For each word in the document, we compute an 100-dimensional vector—we then average these vectors over words in the document, using the averages as features
  - The approach is referred to as CBOW—using CBOW, we replace a layer with $(W + 1)K$ parameters with one with $(B + 1)K$ parameters
  - If W is much less than B (as it typically is), this reduces the number of parameters considerably
  - In Application 1, we have $W = 1,534$, so we reduce the number of parameters by $(W - B)K = 286,800$
  - In Application 2, we have $W = 6,255$, so we reduce the number of parameters by $(W - B)K = 1,231,000$

Deep Learning for Text Analysis

- Does using embeddings help with over-fitting?
  - Using BOW, we have a layer with WK parameters
  - Let's consider using a $B = 100$ dimensional GloVe embedding that was trained on 6 billion tokens
  - For each word in the document, we compute an 100-dimensional vector—we then average these vectors over words in the document, using the averages as features
  - The approach is referred to as CBOW—using CBOW, we replace a layer with $(W + 1)K$ parameters with one with $(B + 1)K$ parameters
  - If W is much less than B (as it typically is), this reduces the number of parameters considerably
  - In Application 1, we have $W = 1,534$, so we reduce the number of parameters by $(W - B)K = 286,800$
  - In Application 2, we have $W = 6,255$, so we reduce the number of parameters by $(W - B)K = 1,231,000$
  - In Application 3, we have $W = 4,868$, so we reduce the number of parameters by $(W - B)K = 953,600$

Deep Learning for Text Analysis

- Does using embeddings help with over-fitting?
  - In Application 1, we were training on 12 million tokens

Deep Learning for Text Analysis

- Does using embeddings help with over-fitting?
  - In Application 1, we were training on 12 million tokens
  - In Application 2, we were training on 29 million tokens
  - In Application 3, we were training on 1 million tokens

Deep Learning for Text Analysis

- Does using embeddings help with over-fitting?
  - In Application 1, we were training on 12 million tokens
  - In Application 2, we were training on 29 million tokens
  - In Application 3, we were training on 1 million tokens
  - Pre-trained embeddings are not specialized to the supervised learning task at hand

Deep Learning for Text Analysis

- Does using embeddings help with over-fitting?
  - In Application 1, we were training on 12 million tokens
  - In Application 2, we were training on 29 million tokens
  - In Application 3, we were training on 1 million tokens
  - Pre-trained embeddings are not specialized to the supervised learning task at hand
  - In our 3 applications, there is far less text data than the 6 billion tokens that our GloVe embeddings were trained on

Deep Learning for Text Analysis

- Application 1: Sentiment in IMDB Reviews

	Mode	Per. Mode	Y=0	Y=1	All
LASSO, BOW:					
Train	0	50.1	88.1	89.8	89.0
Test	1	50.1	85.4	86.6	86.0
NN, BOW, 0 Hidden Layers:					
Train	0	50.1	87.2	90.6	88.9
Test	1	50.1	85.1	87.8	86.5
NN, CBOW, 0 Hidden Layers:					
Train	0	50.1	73.3	81.0	77.1
Test	1	50.1	72.5	81.3	76.9
NN, BOW, 1 Hidden Layer:					
Train	0	50.1	99.8	99.8	99.8
Test	1	50.1	86.5	84.3	85.4
NN, CBOW, 1 Hidden Layer:					
Train	0	50.1	84.4	73.9	79.1
Test	1	50.1	83.3	73.3	78.3
NN, BOW, 2 Hidden Layers:					
Train	0	50.1	99.9	99.4	99.6
Test	1	50.1	87.0	81.6	84.3
NN, CBOW, 2 Hidden Layers:					
Train	0	50.1	79.5	81.0	80.3
Test	1	50.1	78.4	80.1	79.3

Deep Learning for Text Analysis

- Application 1: Sentiment in IMDB Reviews

	Mode	Per. Mode	Y=0	Y=1	All
LASSO, BOW:					
Train	0	50.1	88.1	89.8	89.0
Test	1	50.1	85.4	86.6	86.0
NN, BOW, 0 Hidden Layers:					
Train	0	50.1	87.2	90.6	88.9
Test	1	50.1	85.1	87.8	86.5
NN, CBOW, 0 Hidden Layers:					
Train	0	50.1	73.3	81.0	77.1
Test	1	50.1	72.5	81.3	76.9
NN, BOW, 1 Hidden Layer:					
Train	0	50.1	99.8	99.8	99.8
Test	1	50.1	86.5	84.3	85.4
NN, CBOW, 1 Hidden Layer:					
Train	0	50.1	84.4	73.9	79.1
Test	1	50.1	83.3	73.3	78.3
NN, BOW, 2 Hidden Layers:					
Train	0	50.1	99.9	99.4	99.6
Test	1	50.1	87.0	81.6	84.3
NN, CBOW, 2 Hidden Layers:					
Train	0	50.1	79.5	81.0	80.3
Test	1	50.1	78.4	80.1	79.3

- In this case, CBOW performs worse than BOW

Deep Learning for Text Analysis

- Application 1: Sentiment in IMDB Reviews

	Mode	Per. Mode	Y=0	Y=1	All
LASSO, BOW:					
Train	0	50.1	88.1	89.8	89.0
Test	1	50.1	85.4	86.6	86.0
NN, BOW, 0 Hidden Layers:					
Train	0	50.1	87.2	90.6	88.9
Test	1	50.1	85.1	87.8	86.5
NN, CBOW, 0 Hidden Layers:					
Train	0	50.1	73.3	81.0	77.1
Test	1	50.1	72.5	81.3	76.9
NN, BOW, 1 Hidden Layer:					
Train	0	50.1	99.8	99.8	99.8
Test	1	50.1	86.5	84.3	85.4
NN, CBOW, 1 Hidden Layer:					
Train	0	50.1	84.4	73.9	79.1
Test	1	50.1	83.3	73.3	78.3
NN, BOW, 2 Hidden Layers:					
Train	0	50.1	99.9	99.4	99.6
Test	1	50.1	87.0	81.6	84.3
NN, CBOW, 2 Hidden Layers:					
Train	0	50.1	79.5	81.0	80.3
Test	1	50.1	78.4	80.1	79.3

- In this case, CBOW performs worse than BOW
- CBOW overfits less, but it also fits less!

Deep Learning for Text Analysis

- Application 2: Sentiment in Rotten Tomatoes Reviews

	Mode	Per. Mode	Y=0	Y=1	All
Naive Bayes, BOW:					
Train	1	69.6	66.8	79.4	75.5
Test	1	68.4	62.9	79.3	74.2
LASSO, BOW:					
Train	1	69.6	77.4	94.7	89.5
Test	1	68.4	60.4	84.8	77.1
NN, BOW, 0 Hidden Layers:					
Train	1	69.6	72.1	94.0	87.4
Test	1	68.4	59.4	88.1	79.1
NN, CBOW, 0 Hidden Layers:					
Train	1	69.6	9.5	97.3	70.6
Test	1	68.4	9.3	98.0	70.0
NN, BOW, 1 Hidden Layer:					
Train	1	69.6	95.9	97.9	97.3
Test	1	68.4	60.0	86.9	78.4
NN, CBOW, 1 Hidden Layer:					
Train	1	69.6	18.3	96.3	72.6
Test	1	68.4	17.7	96.3	71.5
NN, BOW, 2 Hidden Layers:					
Train	1	69.6	91.7	99.5	97.1
Test	1	68.4	56.8	88.2	78.3
NN, CBOW, 2 Hidden Layers:					
Train	1	69.6	32.4	91.6	73.6
Test	1	68.4	30.9	91.7	72.5

Deep Learning for Text Analysis

- Application 2: Sentiment in Rotten Tomatoes Reviews

	Mode	Per. Mode	Y=0	Y=1	All
Naive Bayes, BOW:					
Train	1	69.6	66.8	79.4	75.5
Test	1	68.4	62.9	79.3	74.2
LASSO, BOW:					
Train	1	69.6	77.4	94.7	89.5
Test	1	68.4	60.4	84.8	77.1
NN, BOW, 0 Hidden Layers:					
Train	1	69.6	72.1	94.0	87.4
Test	1	68.4	59.4	88.1	79.1
NN, CBOW, 0 Hidden Layers:					
Train	1	69.6	9.5	97.3	70.6
Test	1	68.4	9.3	98.0	70.0
NN, BOW, 1 Hidden Layer:					
Train	1	69.6	95.9	97.9	97.3
Test	1	68.4	60.0	86.9	78.4
NN, CBOW, 1 Hidden Layer:					
Train	1	69.6	18.3	96.3	72.6
Test	1	68.4	17.7	96.3	71.5
NN, BOW, 2 Hidden Layers:					
Train	1	69.6	91.7	99.5	97.1
Test	1	68.4	56.8	88.2	78.3
NN, CBOW, 2 Hidden Layers:					
Train	1	69.6	32.4	91.6	73.6
Test	1	68.4	30.9	91.7	72.5

- We again find that CBOW reduces overfiting, but performs worse

Deep Learning for Text Analysis

- Application 3: Coding whether newspaper articles are discussing the economy

	Mode	Per. Mode	Y=0	Y=1	All
Naive Bayes, BOW:					
Train	1	58.6	78.1	85.9	82.7
Test	1	58.6	72.1	76.2	74.5
LASSO, BOW:					
Train	1	58.6	89.9	94.3	92.5
Test	1	58.6	64.6	72.0	68.9
LASSO, CBOW:					
Train	1	58.6	67.4	84.2	77.3
Test	1	58.6	71.5	79.6	76.3
NN, BOW, 0 Hidden Layers:					
Train	1	58.6	89.3	93.0	91.5
Test	1	58.6	65.9	75.7	71.6
NN, CBOW, 0 Hidden Layers:					
Train	1	58.6	62.0	82.6	74.1
Test	1	58.6	66.2	80.6	74.6
NN, BOW, 1 Hidden Layer:					
Train	1	58.6	95.2	90.5	92.4
Test	1	58.6	68.5	72.0	70.6
NN, CBOW, 1 Hidden Layer:					
Train	1	58.6	68.4	81.8	76.3
Test	1	58.6	71.5	79.4	76.1
NN, BOW, 2 Hidden Layers:					
Train	1	58.6	83.9	98.2	92.3
Test	1	58.6	56.1	80.6	70.4
NN, CBOW, 2 Hidden Layers:					
Train	1	58.6	66.8	86.1	78.1
Test	1	58.6	69.5	81.3	76.4

Deep Learning for Text Analysis

- Application 3: Coding whether newspaper articles are discussing the economy
  - In this case, CBOW leads to improved performance, presumably because the large number of parameters in BOW is more detrimental when the training data is small

Deep Learning for Text Analysis

- Application 3: Coding whether newspaper articles are discussing the economy
  - In this case, CBOW leads to improved performance, presumably because the large number of parameters in BOW is more detrimental when the training data is small
  - CBOW even helps with the LASSO

Deep Learning for Text Analysis

- Application 3: Coding whether newspaper articles are discussing the economy
  - In this case, CBOW leads to improved performance, presumably because the large number of parameters in BOW is more detrimental when the training data is small
  - CBOW even helps with the LASSO
  - The best fitting model is the neural network with CBOW and 2 hidden layers, though the improvement over LASSO with CBOW is so small that it likely due to chance

Deep Learning for Text Analysis

Incorporating the Locations of Words and the Relationships between Words

- Suppose we want to code sentiment in movie reviews whose maximum length is $L = 500$ words and the vocabulary has $W = 5,000$ words

Deep Learning for Text Analysis

Incorporating the Locations of Words and the Relationships between Words

- Suppose we want to code sentiment in movie reviews whose maximum length is $L = 500$ words and the vocabulary has $W = 5,000$ words
- Consider first LASSO (with no interactions) and bag of words ($W = 5,000$ features)
  - Parameters to estimate: $W + 1 = 5,001$ ✓

Deep Learning for Text Analysis

Incorporating the Locations of Words and the Relationships between Words

- Suppose we want to code sentiment in movie reviews whose maximum length is $L = 500$ words and the vocabulary has $W = 5,000$ words
- Consider first LASSO (with no interactions) and bag of words ($W = 5,000$ features)
  - Parameters to estimate: $W + 1 = 5,001$ ✓
  - This does not take into account the location of words

Deep Learning for Text Analysis

Incorporating the Locations of Words and the Relationships between Words

- Suppose we want to code sentiment in movie reviews whose maximum length is $L = 500$ words and the vocabulary has $W = 5,000$ words
- Consider first LASSO (with no interactions) and bag of words ($W = 5,000$ features)
  - Parameters to estimate: $W + 1 = 5,001$ ✓
  - This does not take into account the location of words
- Consider LASSO and one-hot encoding for each word ($LW = 2,500,000$ features)

Deep Learning for Text Analysis

Incorporating the Locations of Words and the Relationships between Words

- Suppose we want to code sentiment in movie reviews whose maximum length is $L = 500$ words and the vocabulary has $W = 5,000$ words
- Consider first LASSO (with no interactions) and bag of words ($W = 5,000$ features)
  - Parameters to estimate: $W + 1 = 5,001$ ✓
  - This does not take into account the location of words
- Consider LASSO and one-hot encoding for each word ($LW = 2,500,000$ features)
  - Parameters to estimate: $LW + 1 = 2,500,001$ ✗

Deep Learning for Text Analysis

Incorporating the Locations of Words and the Relationships between Words

- Suppose we want to code sentiment in movie reviews whose maximum length is $L = 500$ words and the vocabulary has $W = 5,000$ words
- Consider first LASSO (with no interactions) and bag of words ($W = 5,000$ features)
  - Parameters to estimate: $W + 1 = 5,001$ ✓
  - This does not take into account the location of words
- Consider LASSO and one-hot encoding for each word ($LW = 2,500,000$ features)
  - Parameters to estimate: $LW + 1 = 2,500,001$ ✗
  - This takes into account the location of words, but not relationships between them

Deep Learning for Text Analysis

- Consider instead a neural network with two fully connected layers, each with $K = 100$ neurons, and $J = 2$ outcomes

Deep Learning for Text Analysis

- Consider instead a neural network with two fully connected layers, each with $K = 100$ neurons, and $J = 2$ outcomes
  - Parameters to estimate:
    $(LW + 1)K + (K + 1)K + (K + 1)J =$
    $250,000,100 + 10,100 + 202 = 250,010,402$ ✗

Deep Learning for Text Analysis

- Consider instead a neural network with two fully connected layers, each with $K = 100$ neurons, and $J = 2$ outcomes
  - Parameters to estimate:
    $(LW + 1)K + (K + 1)K + (K + 1)J =$
    $250,000,100 + 10,100 + 202 = 250,010,402$ ✗
  - Potentially accounts for word relationships, but too many parameters!

Deep Learning for Text Analysis

- Consider instead a neural network with two fully connected layers, each with $K = 100$ neurons, and $J = 2$ outcomes
  - Parameters to estimate:
    $(LW + 1)K + (K + 1)K + (K + 1)J =$
    $250,000,100 + 10,100 + 202 = 250,010,402$ ✗
  - Potentially accounts for word relationships, but too many parameters!
- Alternatively, we could consider bag of words, where the features sum the one hot encoding over the words in the document

Deep Learning for Text Analysis

- Consider instead a neural network with two fully connected layers, each with $K = 100$ neurons, and $J = 2$ outcomes
  - Parameters to estimate:
    $(LW + 1)K + (K + 1)K + (K + 1)J =$
    $250,000,100 + 10,100 + 202 = 250,010,402$ ✗
  - Potentially accounts for word relationships, but too many parameters!
- Alternatively, we could consider bag of words, where the features sum the one hot encoding over the words in the document
  - Parameters to estimate: $(W + 1)K + (K + 1)K + (K + 1)J =$
    $500,100 + 10,100 + 202 = 510,402$ ✓

Deep Learning for Text Analysis

- Consider instead a neural network with two fully connected layers, each with $K = 100$ neurons, and $J = 2$ outcomes
  - Parameters to estimate:
    $(LW + 1)K + (K + 1)K + (K + 1)J =$
    $250,000,100 + 10,100 + 202 = 250,010,402$ ✗
  - Potentially accounts for word relationships, but too many parameters!
- Alternatively, we could consider bag of words, where the features sum the one hot encoding over the words in the document
  - Parameters to estimate: $(W + 1)K + (K + 1)K + (K + 1)J =$
    $500,100 + 10,100 + 202 = 510,402$ ✓
  - No longer accounts for relationship between words and instead accounts for relationship between word sums

Deep Learning for Text Analysis

- We could consider a common mapping of the one-hot encoding to word vectors of size $B = 100$

Deep Learning for Text Analysis

- We could consider a common mapping of the one-hot encoding to word vectors of size $B = 100$
  - Parameters to estimate:
    $(W+1)B + (LB+1)K + (K+1)K + (K+1)J =$
    $500,100 + 5,000,100 + 10,100 + 202 = 5,510,502$ ✗

Deep Learning for Text Analysis

- We could consider a common mapping of the one-hot encoding to word vectors of size $B = 100$
  - Parameters to estimate:
    $(W+1)B + (LB+1)K + (K+1)K + (K+1)J =$
    $500,100 + 5,000,100 + 10,100 + 202 = 5,510,502$ ✗
- We could use pre-trained word embeddings
  - Parameters to estimate: $(LB+1)K + (K+1)K + (K+1)J =$
    $5,000,100 + 10,100 + 202 = 5,010,402$ ✗

Deep Learning for Text Analysis

- We could consider a common mapping of the one-hot encoding to word vectors of size $B = 100$
  - Parameters to estimate:
    $(W+1)B + (LB+1)K + (K+1)K + (K+1)J =$
    $500,100 + 5,000,100 + 10,100 + 202 = 5,510,502$ ✗
- We could use pre-trained word embeddings
  - Parameters to estimate: $(LB+1)K + (K+1)K + (K+1)J =$
    $5,000,100 + 10,100 + 202 = 5,010,402$ ✗
- We could use continuous bag of words with a common map
  - Parameters to estimate:
    $(W+1)B + (B+1)K + (K+1)K + (K+1)J =$
    $500,100 + 10,100 + 10,100 + 202 = 520,502$ ✓

Deep Learning for Text Analysis

- We could use continuous bag of words with pre-trained embeddings
  - Parameters to estimate: $(B+1)K + (K+1)K + (K+1)J = 10,100 + 10,100 + 202 = 20,402$ ✓

Deep Learning for Text Analysis

- We could use continuous bag of words with pre-trained embeddings
  - Parameters to estimate: $(B+1)K + (K+1)K + (K+1)J = 10,100 + 10,100 + 202 = 20,402$ ✓
  - Were back to the issue that we are not accounting for relationships between words

Deep Learning for Text Analysis

- Other values for L, W, and B:

L (context length)	500	200	20
W (vocab. size)	5000	5000	5000
B (embedding dim.)	100	50	50
K (hidden layer size)	100	100	100
J (output size)	2	2	2
Lasso	2,500,001✗	1,000,001✓	100,001✓
BOW	5,001✓	5,001✓	5,001✓
NN, 2 Layers	250,010,402✗	100,010,402✗	10,010,402✗
BOW	510,402✓	510,402✓	510,402✓
common map	5,510,502 ✗	1,260,452 ✓	360,452✓
common map, pre-trained emb.	5,010,402✗	1,010,402 ✓	110,402✓
CBOW	520,502 ✓	265,452 ✓	265,452✓
CBOW, pre-trained emb.	20,402 ✓	15,402 ✓	15,402✓

Deep Learning for Text Analysis

Recurrent Neural Networks

- BOW and CBOW with pre-trained embedding are feasible, but do not take into account the locations and relationships between words

Deep Learning for Text Analysis

Recurrent Neural Networks

- BOW and CBOW with pre-trained embedding are feasible, but do not take into account the locations and relationships between words
- Approaches that take into account locations involve excessive parameters, even when pre-trained embeddings are used (except when sequence length is short)

Deep Learning for Text Analysis

Recurrent Neural Networks

- BOW and CBOW with pre-trained embedding are feasible, but do not take into account the locations and relationships between words
- Approaches that take into account locations involve excessive parameters, even when pre-trained embeddings are used (except when sequence length is short)
- The problem above was that to preserve the locations of words, we need a map between LW features (LB features with embeddings) to K neurons, which resulted in excessive parameters

Deep Learning for Text Analysis

Recurrent Neural Networks

- BOW and CBOW with pre-trained embedding are feasible, but do not take into account the locations and relationships between words
- Approaches that take into account locations involve excessive parameters, even when pre-trained embeddings are used (except when sequence length is short)
- The problem above was that to preserve the locations of words, we need a map between LW features (LB features with embeddings) to K neurons, which resulted in excessive parameters
- This approach essentially allows for a separate map from each location to the neurons

Deep Learning for Text Analysis

Recurrent Neural Networks

- We aim to introduce a common map between locations and neurons, while preserving the ability to account for locations of words

Deep Learning for Text Analysis

Recurrent Neural Networks

- We aim to introduce a common map between locations and neurons, while preserving the ability to account for locations of words
- Recurrent Neural Networks (RNNs) meet this requirement

Deep Learning for Text Analysis

Recurrent Neural Networks

- Elman Network:
  - $h_t = g_1(A_{11}x_t + A_{12}h_{t-1} + b_1)$
  - $y_T = g_2(A_2 h_T + b_2)$

Deep Learning for Text Analysis

Recurrent Neural Networks

- Elman Network:
  - $h_t = g_1(A_{11}x_t + A_{12}h_{t-1} + b_1)$
  - $y_T = g_2(A_2 h_T + b_2)$
- Number of parameters:
  $(B + K + 1)K + (K + 1)J = 20,100 + 202 = 20,302$ ✓

Deep Learning for Text Analysis

Recurrent Neural Networks

- Elman Network:
  - $h_t = g_1(A_{11}x_t + A_{12}h_{t-1} + b_1)$
  - $y_T = g_2(A_2 h_T + b_2)$
- Number of parameters:
  $(B + K + 1)K + (K + 1)J = 20,100 + 202 = 20,302$ ✓
- h_t are hidden layers, which essentially act as multiple weighted running tally's as we move through the document

Deep Learning for Text Analysis

Recurrent Neural Networks

- Elman Network:
  - $h_t = g_1(A_{11}x_t + A_{12}h_{t-1} + b_1)$
  - $y_T = g_2(A_2 h_T + b_2)$
- Number of parameters:
  $(B + K + 1)K + (K + 1)J = 20,100 + 202 = 20,302$ ✓
- h_t are hidden layers, which essentially act as multiple weighted running tally's as we move through the document
- The multiple running tallies act to allow for relationships between words to affect the prediction

Deep Learning for Text Analysis

Recurrent Neural Networks

- Elman Network:
  - $h_t = g_1(A_{11}x_t + A_{12}h_{t-1} + b_1)$
  - $y_T = g_2(A_2 h_T + b_2)$
- Number of parameters:
  $(B + K + 1)K + (K + 1)J = 20,100 + 202 = 20,302$ ✓
- h_t are hidden layers, which essentially act as multiple weighted running tally's as we move through the document
- The multiple running tallies act to allow for relationships between words to affect the prediction
- The fact that A_{11}, A_{12}, and b_1 are common limit the number of parameters

Deep Learning for Text Analysis

Recurrent Neural Networks

- RNNs have a second benefit—they potentially work on variable length inputs (e.g. text documents)

Deep Learning for Text Analysis

Recurrent Neural Networks

- RNNs have a second benefit—they potentially work on variable length inputs (e.g. text documents)
- Despite this, it is often common practice to set a maximum document length and pad shorter documents, effectively not employing one of the benefits of RNNs

Deep Learning for Text Analysis

Recurrent Neural Networks

- Elman networks have difficulty focusing on words that are far apart

Deep Learning for Text Analysis

Recurrent Neural Networks

- Elman networks have difficulty focusing on words that are far apart
- An alternative RNN is called the Long Short-term Memory (LSTM) network:

Deep Learning for Text Analysis

Recurrent Neural Networks

- Elman networks have difficulty focusing on words that are far apart
- An alternative RNN is called the Long Short-term Memory (LSTM) network:
  - $f_t = g_1(A_{11}x_t + A_{12}h_{t-1} + b_1)$
  - $i_t = g_1(A_{21}x_t + A_{22}h_{t-1} + b_2)$
  - $o_t = g_1(A_{31}x_t + A_{23}h_{t-1} + b_3)$
  - $z_t = g_2(A_{41}x_t + A_{24}h_{t-1} + b_4)$
  - $c_t = f_t \circ c_{t-1} + i_t \circ z_t$
  - $h_t = o_t \circ g_3(c_t)$
  - $y_T = g_1(A_5 h_T + b_5)$

  where $\circ$ is the element-wise product

Deep Learning for Text Analysis

Recurrent Neural Networks

- Intuition for LSTM:
  - c_t potentially remembers the past and z_t is new information

Deep Learning for Text Analysis

Recurrent Neural Networks

- Intuition for LSTM:
  - c_t potentially remembers the past and z_t is new information
  - f_t controls whether we forget and i_t controls whether new information flows through, potentially accounting for relationships between words that are farther apart

Deep Learning for Text Analysis

Recurrent Neural Networks

- Intuition for LSTM:
  - c_t potentially remembers the past and z_t is new information
  - f_t controls whether we forget and i_t controls whether new information flows through, potentially accounting for relationships between words that are farther apart
  - Well, intuition is not perfect! (i.e. why the formula for h_t?)

Deep Learning for Text Analysis

Recurrent Neural Networks

- Intuition for LSTM:
  - c_t potentially remembers the past and z_t is new information
  - f_t controls whether we forget and i_t controls whether new information flows through, potentially accounting for relationships between words that are farther apart
  - Well, intuition is not perfect! (i.e. why the formula for h_t?)
  - As usual, the formulation exists because it "works"

Deep Learning for Text Analysis

Recurrent Neural Networks

- Intuition for LSTM:
  - c_t potentially remembers the past and z_t is new information
  - f_t controls whether we forget and i_t controls whether new information flows through, potentially accounting for relationships between words that are farther apart
  - Well, intuition is not perfect! (i.e. why the formula for h_t?)
  - As usual, the formulation exists because it "works"
  - Counting parameters,
    $WB + 4(B + K + 1)K + (K + 1)J = 580,602$ ✓ (Pytorch weirdly implements LSTM with double intercepts, so it's actually $WB + 4(B + K + 2)K + (K + 1)J = 581,002$ ✓)

Deep Learning for Text Analysis

Recurrent Neural Networks

- Intuition for LSTM:
  - c_t potentially remembers the past and z_t is new information
  - f_t controls whether we forget and i_t controls whether new information flows through, potentially accounting for relationships between words that are farther apart
  - Well, intuition is not perfect! (i.e. why the formula for h_t?)
  - As usual, the formulation exists because it "works"
  - Counting parameters,
    $WB + 4(B + K + 1)K + (K + 1)J = 580,602$ ✓ (Pytorch weirdly implements LSTM with double intercepts, so it's actually $WB + 4(B + K + 2)K + (K + 1)J = 581,002$ ✓)
  - The number of parameters does not increase with the sequence length

Deep Learning for Text Analysis

Recurrent Neural Networks

- Intuition for LSTM:
  - c_t potentially remembers the past and z_t is new information
  - f_t controls whether we forget and i_t controls whether new information flows through, potentially accounting for relationships between words that are farther apart
  - Well, intuition is not perfect! (i.e. why the formula for h_t?)
  - As usual, the formulation exists because it "works"
  - Counting parameters, $WB + 4(B + K + 1)K + (K + 1)J = 580,602$ ✓ (Pytorch weirdly implements LSTM with double intercepts, so it's actually $WB + 4(B + K + 2)K + (K + 1)J = 581,002$ ✓)
  - The number of parameters does not increase with the sequence length
  - With pre-trained embeddings, the number of parameters would be $4(B + K + 1)K + (K + 1)J = 80,602$ ✓ (though it is also common to use a smaller B when the embeddings are trained)

Deep Learning for Text Analysis

- Feasibility of LSTM:

L (context length)	500	200	20
W (vocab. size)	5000	5000	5000
B (embedding dim.)	100	50	50
K (hidden layer size)	100	100	100
J (output size)	2	2	2
Lasso	2,500,001✗	1,000,001✓	100,001✓
BOW	5,001✓	5,001✓	5,001✓
NN, 2 Layers	250,010,402✗	100,010,402✗	10,010,402✗
BOW	510,402✓	510,402✓	510,402✓
common map	5,510,502 ✗	1,260,452 ✓	360,452✓
common map, pre-trained emb.	5,010,402✗	1,010,402 ✓	110,402✓
CBOW	520,502 ✓	265,452 ✓	265,452✓
CBOW, pre-trained emb.	20,402 ✓	15,402 ✓	15,402✓
LSTM	580,602✓	310,602 ✓	310,602✓

Deep Learning for Text Analysis

Transformer Models

- Now for something absurdly complicated...

Deep Learning for Text Analysis

Transformer Models

- Now for something absurdly complicated...
- LSTM models allow words to affect each other, but in a limited way

Deep Learning for Text Analysis

Transformer Models

- Now for something absurdly complicated...
- LSTM models allow words to affect each other, but in a limited way
- Transformer Models allow each word to affect every other word

Deep Learning for Text Analysis

Transformer Models

- Let's embed a sequence of L words using a B-dimensional embedding, with the matrix $\underset{L \times B}{E}$

Deep Learning for Text Analysis

Transformer Models

- Let's embed a sequence of L words using a B-dimensional embedding, with the matrix $\underset{L \times B}{E}$
- Let's imagine weighting every word based on every other word:

$$\underset{LB \times LB}{W} \quad \underset{LB \times 1}{vec(E)}$$

Deep Learning for Text Analysis

Transformer Models

- Let's embed a sequence of L words using a B-dimensional embedding, with the matrix $\underset{L\times B}{E}$
- Let's imagine weighting every word based on every other word:

$$\underset{LB\times LB}{W}\ \underset{LB\times 1}{vec(E)}$$

- We can have somewhat fewer parameters as follows:

$$\underset{L\times L}{W}\ \underset{L\times B}{E}$$

where we assume that $W_{ij} \geq 0$ and $\sum_{j=1}^{L} W_{ij} = 1$.

Deep Learning for Text Analysis

Transformer Models

- We can implement this as, $W_{ij} = \frac{e^{Z_{ij}}}{\sum_{k=1}^{L} e^{Z_{ik}}}$

Deep Learning for Text Analysis

Transformer Models

- We can implement this as, $W_{ij} = \frac{e^{Z_{ij}}}{\sum_{k=1}^{L} e^{Z_{ik}}}$
- We can reduce parameters as follows:

$$\underset{L\times L}{Z} = \underset{L\times B}{E}\ \underset{B\times B}{\Omega}\ \underset{B\times L}{E}$$

where $W = \Lambda(Z) = \Lambda(E\Omega E')$ is called the attention matrix and Λ is a softmax

Deep Learning for Text Analysis

Transformer Models

- We can implement this as, $W_{ij} = \frac{e^{Z_{ij}}}{\sum_{k=1}^{L} e^{Z_{ik}}}$
- We can reduce parameters as follows:

$$\underset{L\times L}{Z} = \underset{L\times B}{E}\ \underset{B\times B}{\Omega}\ \underset{B\times L}{E}$$

where $W = \Lambda(Z) = \Lambda(E\Omega E')$ is called the attention matrix and Λ is a softmax

- As E enters into the attention placed on E, this is referred to as self attention

Deep Learning for Text Analysis

Transformer Models

- We can implement this as, $W_{ij} = \frac{e^{Z_{ij}}}{\sum_{k=1}^{L} e^{Z_{ik}}}$
- We can reduce parameters as follows:

$$\underset{L\times L}{Z} = \underset{L\times B}{E}\ \underset{B\times B}{\Omega}\ \underset{B\times L}{E}$$

where $W = \Lambda(Z) = \Lambda(E\Omega E')$ is called the attention matrix and Λ is a softmax

- As E enters into the attention placed on E, this is referred to as self attention
- We often will use multiple attention matrices (called heads):

$$\underset{L\times B}{R} = \sum_{h=1}^{H} \left(\underset{L\times L}{\Lambda(E\Omega_h E')}\ \underset{L\times B}{E}\ \underset{B\times B}{W_h^V} \right) \underset{B\times B}{W^O}$$

Deep Learning for Text Analysis

Transformer Models

- It is common to further over-parameterize the model as:

$$\underset{L\times B}{R} = \sum_{h=1}^{H} \left(\Lambda(\underset{L\times B}{E} \underset{B\times B}{W_h^Q} (\underset{B\times B}{W_h^K})' \underset{B\times L}{E'}) \underset{L\times B}{E} \underset{B\times B}{W_h^V} \right) \underset{B\times B}{W^O}$$

Deep Learning for Text Analysis

Transformer Models

- It is common to further over-parameterize the model as:

$$\underset{L\times B}{R} = \sum_{h=1}^{H} \left(\Lambda(\underset{L\times B}{E} \underset{B\times B}{W_h^Q} (\underset{B\times B}{W_h^K})' \underset{B\times L}{E'}) \underset{L\times B}{E} \underset{B\times B}{W_h^V} \right) \underset{B\times B}{W^O}$$

- Note that the dimensions of both the input (E) and the output (R) are $L \times B$, so transformer layers can be stacked multiple times

Deep Learning for Text Analysis

Transformer Models

- It is common to further over-parameterize the model as:

$$\underset{L\times B}{R} = \sum_{h=1}^{H} \left(\Lambda(\underset{L\times B}{E} \underset{B\times B}{W_h^Q} (\underset{B\times B}{W_h^K})' \underset{B\times L}{E'}) \underset{L\times B}{E} \underset{B\times B}{W_h^V} \right) \underset{B\times B}{W^O}$$

- Note that the dimensions of both the input (E) and the output (R) are $L \times B$, so transformer layers can be stacked multiple times
- The final output of transformer layers, $\underset{L\times B}{R}$, is averaged across words, to produce $\underset{1\times B}{r}$, which is then fed through a fully connected layer with RELU

Deep Learning for Text Analysis

Transformer Models

- If $H = 3$, counting parameters, we have $WB + 3HB(B+1) + (3B+1)B + (B+1)K + (K+1)J = 631,302$ ✓

Deep Learning for Text Analysis

Transformer Models

- If $H = 3$, counting parameters, we have $WB + 3HB(B + 1) + (3B + 1)B + (B + 1)K + (K + 1)J = 631,302$ ✓
- The number of parameters does not increase with the sequence length

Deep Learning for Text Analysis

Transformer Models

- If $H = 3$, counting parameters, we have $WB + 3HB(B+1) + (3B+1)B + (B+1)K + (K+1)J = 631,302$ ✓
- The number of parameters does not increase with the sequence length
- With pre-trained embeddings, the number of parameters would be $3HB(B+1) + (3B+1)B + (B+1)K + (K+1)J = 131,302$ ✓ (and like LSTM, it is common to use a smaller B when the embeddings are trained)

Deep Learning for Text Analysis

Transformer Models

- Other things that are included in the model: Positional encoding:
  - The position of a word is encoded similarly to words—in a B dimensional space—this introduces an additional BL parameters to the model

Deep Learning for Text Analysis

Transformer Models

- Other things that are included in the model: Positional encoding:
  - The position of a word is encoded similarly to words—in a B dimensional space—this introduces an additional BL parameters to the model
  - Things get weirder: the word embedding and the positional embedding are added to each other

Deep Learning for Text Analysis

Transformer Models

- Other things that are included in the model: Positional encoding:
  - The position of a word is encoded similarly to words—in a B dimensional space—this introduces an additional BL parameters to the model
  - Things get weirder: the word embedding and the positional embedding are added to each other
- Masking is employed to essentially ignore the padded words in the calculation—if $L = 500$ and the input only had 243 words, the last 257 words would be masked (no new parameters introduced)

Deep Learning for Text Analysis

Transformer Models

- Other things that are included in the model: Positional encoding:
  - The position of a word is encoded similarly to words—in a B dimensional space—this introduces an additional BL parameters to the model
  - Things get weirder: the word embedding and the positional embedding are added to each other
- Masking is employed to essentially ignore the padded words in the calculation—if $L = 500$ and the input only had 243 words, the last 257 words would be masked (no new parameters introduced)
- Dropout is used after self-attention (no new parameters)

Deep Learning for Text Analysis

Transformer Models

- A residual connection is employed—if x is the input to the attention block and y be the output after dropout, x and y are added together (no new parameters)

Deep Learning for Text Analysis

Transformer Models

- A residual connection is employed—if x is the input to the attention block and y be the output after dropout, x and y are added together (no new parameters)
- Layer normalization is applied to the sum, setting the mean equal to 0 and the variance equal to 1 (adds $2B$ parameters)

Deep Learning for Text Analysis

Transformer Models

- A residual connection is employed—if x is the input to the attention block and y be the output after dropout, x and y are added together (no new parameters)
- Layer normalization is applied to the sum, setting the mean equal to 0 and the variance equal to 1 (adds $2B$ parameters)
- An inner layer is used within the transformer, with I neurons (adds $(I+1)B + (B+1)I$ parameters)

Deep Learning for Text Analysis

Transformer Models

- A residual connection is employed—if x is the input to the attention block and y be the output after dropout, x and y are added together (no new parameters)
- Layer normalization is applied to the sum, setting the mean equal to 0 and the variance equal to 1 (adds $2B$ parameters)
- An inner layer is used within the transformer, with I neurons (adds $(I+1)B + (B+1)I$ parameters)
- Layer normalization is used on the inner layers (adds $2I$ parameters)

Deep Learning for Text Analysis

Transformer Models

- A residual connection is employed—if x is the input to the attention block and y be the output after dropout, x and y are added together (no new parameters)
- Layer normalization is applied to the sum, setting the mean equal to 0 and the variance equal to 1 (adds $2B$ parameters)
- An inner layer is used within the transformer, with I neurons (adds $(I+1)B + (B+1)I$ parameters)
- Layer normalization is used on the inner layers (adds $2I$ parameters)
- An outer fully connected layer is used, adding $(B+1)K$ parameters

Deep Learning for Text Analysis

Transformer Models

- Overall parameter count:

Word Embedding	WB
Positional Embedding	LB
Transformer	$3HB(B+1)+(3B+1)B$
Trans., Inner FC Layer	$(B+1)I+(I+1)B$
Layer Normalization	$2B+2I$
Outer FC Layer	$(B+1)K$
Output	$(K+1)J$

Deep Learning for Text Analysis

- Application 1: Sentiment in IMDB Reviews

	Mode	Per. Mode	Y=0	Y=1	All
LASSO, BOW:					
Train	0	50.1	88.1	89.8	89.0
Test	1	50.1	85.4	86.6	86.0
NN, BOW, 0 Hidden Layers:					
Train	0	50.1	87.2	90.6	88.9
Test	1	50.1	85.1	87.8	86.5
NN, BOW, 1 Hidden Layer:					
Train	0	50.1	99.8	99.8	99.8
Test	1	50.1	86.5	84.3	85.4
NN, BOW, 2 Hidden Layers:					
Train	0	50.1	99.9	99.4	99.6
Test	1	50.1	87.0	81.6	84.3
LSTM:					
Train	0	50.1	94.3	86.0	90.2
Test	1	50.1	90.7	81.2	85.9
Transformer:					
Train	0	50.1	93.6	88.5	91.0
Test	0	50.1	90.2	83.6	86.9

Deep Learning for Text Analysis

- Application 1: Sentiment in IMDB Reviews

	Mode	Per. Mode	Y=0	Y=1	All
LASSO, BOW:					
Train	0	50.1	88.1	89.8	89.0
Test	1	50.1	85.4	86.6	86.0
NN, BOW, 0 Hidden Layers:					
Train	0	50.1	87.2	90.6	88.9
Test	1	50.1	85.1	87.8	86.5
NN, BOW, 1 Hidden Layer:					
Train	0	50.1	99.8	99.8	99.8
Test	1	50.1	86.5	84.3	85.4
NN, BOW, 2 Hidden Layers:					
Train	0	50.1	99.9	99.4	99.6
Test	1	50.1	87.0	81.6	84.3
LSTM:					
Train	0	50.1	94.3	86.0	90.2
Test	1	50.1	90.7	81.2	85.9
Transformer:					
Train	0	50.1	93.6	88.5	91.0
Test	0	50.1	90.2	83.6	86.9

- The transformer model is the best fitting model, by a very small amount

Deep Learning for Text Analysis

- Application 2: Sentiment in Rotten Tomatoes Reviews

	Mode	Per. Mode	Y=0	Y=1	All
Naive Bayes, BOW:					
Train	1	69.6	66.8	79.4	75.5
Test	1	68.4	62.9	79.3	74.2
LASSO, BOW:					
Train	1	69.6	77.4	94.7	89.5
Test	1	68.4	60.4	84.8	77.1
NN, BOW, 0 Hidden Layers:					
Train	1	69.6	72.1	94.0	87.4
Test	1	68.4	59.4	88.1	79.1
NN, BOW, 1 Hidden Layer:					
Train	1	69.6	95.9	97.9	97.3
Test	1	68.4	60.0	86.9	78.4
NN, BOW, 2 Hidden Layers:					
Train	1	69.6	91.7	99.5	97.1
Test	1	68.4	56.8	88.2	78.3
LSTM:					
Train	1	69.6	70.3	92.4	85.7
Test	1	68.4	56.3	88.6	78.4
Transformer:					
Train	1	69.6	78.4	91.2	87.3
Test	1	68.4	64.8	84.6	78.3

Deep Learning for Text Analysis

- Application 2: Sentiment in Rotten Tomatoes Reviews

	Mode	Per. Mode	Y=0	Y=1	All
Naive Bayes, BOW:					
Train	1	69.6	66.8	79.4	75.5
Test	1	68.4	62.9	79.3	74.2
LASSO, BOW:					
Train	1	69.6	77.4	94.7	89.5
Test	1	68.4	60.4	84.8	77.1
NN, BOW, 0 Hidden Layers:					
Train	1	69.6	72.1	94.0	87.4
Test	1	68.4	59.4	88.1	79.1
NN, BOW, 1 Hidden Layer:					
Train	1	69.6	95.9	97.9	97.3
Test	1	68.4	60.0	86.9	78.4
NN, BOW, 2 Hidden Layers:					
Train	1	69.6	91.7	99.5	97.1
Test	1	68.4	56.8	88.2	78.3
LSTM:					
Train	1	69.6	70.3	92.4	85.7
Test	1	68.4	56.3	88.6	78.4
Transformer:					
Train	1	69.6	78.4	91.2	87.3
Test	1	68.4	64.8	84.6	78.3

- The neural network with BOW and 0 hidden layers remains the best fitting model

Deep Learning for Text Analysis

- Application 3: Coding whether newspaper articles are discussing the economy

	Mode	Per. Mode	Y=0	Y=1	All
Naive Bayes, BOW:					
Train	1	58.6	78.1	85.9	82.7
Test	1	58.6	72.1	76.2	74.5
LASSO, BOW:					
Train	1	58.6	89.9	94.3	92.5
Test	1	58.6	64.6	72.0	68.9
LASSO, CBOW:					
Train	1	58.6	67.4	84.2	77.3
Test	1	58.6	71.5	79.6	76.3
NN, CBOW, 0 Hidden Layers:					
Train	1	58.6	62.0	82.6	74.1
Test	1	58.6	66.2	80.6	74.6
NN, CBOW, 1 Hidden Layer:					
Train	1	58.6	68.4	81.8	76.3
Test	1	58.6	71.5	79.4	76.1
NN, CBOW, 2 Hidden Layers:					
Train	1	58.6	66.8	86.1	78.1
Test	1	58.6	69.5	81.3	76.4
LSTM, Pre. Emb.:					
Train	1	58.6	72.0	81.4	77.5
Test	1	58.6	74.4	76.0	75.3
Transformer, Pre. Emb.:					
Train	1	58.6	44.9	94.9	74.2
Test	1	58.6	46.6	93.8	74.2

Deep Learning for Text Analysis

- Application 3: Coding whether newspaper articles are discussing the economy

	Mode	Per. Mode	Y=0	Y=1	All
Naive Bayes, BOW:					
Train	1	58.6	78.1	85.9	82.7
Test	1	58.6	72.1	76.2	74.5
LASSO, BOW:					
Train	1	58.6	89.9	94.3	92.5
Test	1	58.6	64.6	72.0	68.9
LASSO, CBOW:					
Train	1	58.6	67.4	84.2	77.3
Test	1	58.6	71.5	79.6	76.3
NN, CBOW, 0 Hidden Layers:					
Train	1	58.6	62.0	82.6	74.1
Test	1	58.6	66.2	80.6	74.6
NN, CBOW, 1 Hidden Layer:					
Train	1	58.6	68.4	81.8	76.3
Test	1	58.6	71.5	79.4	76.1
NN, CBOW, 2 Hidden Layers:					
Train	1	58.6	66.8	86.1	78.1
Test	1	58.6	69.5	81.3	76.4
LSTM, Pre. Emb.:					
Train	1	58.6	72.0	81.4	77.5
Test	1	58.6	74.4	76.0	75.3
Transformer, Pre. Emb.:					
Train	1	58.6	44.9	94.9	74.2
Test	1	58.6	46.6	93.8	74.2

- Neural network with CBOW remains the best fitting model

Deep Learning for Text Analysis

Pretrained Models

- Models that considered the locations and relationships between words did not lead to substantial improvements in fit

Deep Learning for Text Analysis

Pretrained Models

- Models that considered the locations and relationships between words did not lead to substantial improvements in fit
- This may be because models and data sets have to be very big to take full advantage of these models

Deep Learning for Text Analysis

Pretrained Models

- Models that considered the locations and relationships between words did not lead to substantial improvements in fit
- This may be because models and data sets have to be very big to take full advantage of these models
- Let's consider one such big model, RoBERTa (Liu et al., 2019)—I'll call this big model a medium size model to distinguish for the models we just implemented, though the literature now refers to this a small model

Deep Learning for Text Analysis

Pretrained Models

- Models that considered the locations and relationships between words did not lead to substantial improvements in fit
- This may be because models and data sets have to be very big to take full advantage of these models
- Let's consider one such big model, RoBERTa (Liu et al., 2019)—I'll call this big model a medium size model to distinguish for the models we just implemented, though the literature now refers to this a small model
- Trained on 160GB of text (for comparison, Application 1 was 64MB, Application 2 was 180MB, and Application 3 was 15MB)

Deep Learning for Text Analysis

Pretrained Models

- Models that considered the locations and relationships between words did not lead to substantial improvements in fit
- This may be because models and data sets have to be very big to take full advantage of these models
- Let's consider one such big model, RoBERTa (Liu et al., 2019)—I'll call this big model a medium size model to distinguish for the models we just implemented, though the literature now refers to this a small model
- Trained on 160GB of text (for comparison, Application 1 was 64MB, Application 2 was 180MB, and Application 3 was 15MB)
    - The text includes BookCorpus (free novels written by unpublished authors), English Wikipedia, CC-NEWS (a large number of English news articles from the web), OpenWebText (a collection of webpages shared on reddit), and Stories (story-like text downloaded from the internet)

Deep Learning for Text Analysis

Pretrained Models

- The model has 335M parameters (for comparison, the biggest model we estimated had 1.3 million parameters)

Deep Learning for Text Analysis

Pretrained Models

- The model has 335M parameters (for comparison, the biggest model we estimated had 1.3 million parameters)
  - Model was trained used 1024 GPUs!

Deep Learning for Text Analysis

Pretrained Models

- The model has 335M parameters (for comparison, the biggest model we estimated had 1.3 million parameters)
  - Model was trained used 1024 GPUs!
  - The amount of data rather than the number of parameters is the computational bottleneck

Deep Learning for Text Analysis

Pretrained Models

- The RoBERTa model is based on the BERT model (Devlin et al., 2019), where BERT stands for Bidirectional Encoder Representations from Transformers, which is itself based on Vaswani et al. (2017)

Deep Learning for Text Analysis

Pretrained Models

- The RoBERTa model is based on the BERT model (Devlin et al., 2019), where BERT stands for Bidirectional Encoder Representations from Transformers, which is itself based on Vaswani et al. (2017)
  - Step 1: Tokenization — convert the document to 'words', but it a more sophisticated way

Deep Learning for Text Analysis

Pretrained Models

- The RoBERTa model is based on the BERT model (Devlin et al., 2019), where BERT stands for Bidirectional Encoder Representations from Transformers, which is itself based on Vaswani et al. (2017)
  - Step 1: Tokenization — convert the document to 'words', but it a more sophisticated way
  - Step 2: Unsupervised learning (or learning without labeled data)

Deep Learning for Text Analysis

Pretrained Models

- The RoBERTa model is based on the BERT model (Devlin et al., 2019), where BERT stands for Bidirectional Encoder Representations from Transformers, which is itself based on Vaswani et al. (2017)
  - Step 1: Tokenization — convert the document to 'words', but it a more sophisticated way
  - Step 2: Unsupervised learning (or learning without labeled data)
  - Step 3: Fine-tunning using supervised learning on a number of tasks (learning with labeled data)

Deep Learning for Text Analysis

Tokenization

- Thus far, we have treated the raw inputs into our models as words—this can lead to a loss of information—consider the sentence: "The economy grew by 6.32% in the first quater, a fact highlighted by Scott Bessent"

Deep Learning for Text Analysis

Tokenization

- Thus far, we have treated the raw inputs into our models as words—this can lead to a loss of information—consider the sentence: "The economy grew by 6.32% in the first quater, a fact highlighted by Scott Bessent"
- This might be tokenized as, ('The',' economy',' grew',' by',' ','6','.','32','%',' in',' the',' first',' qu','ater',',',' a',' fact',' highlighted',' by',' Scott',' B','ess','ent')

Deep Learning for Text Analysis

Tokenization

- Thus far, we have treated the raw inputs into our models as words—this can lead to a loss of information—consider the sentence: "The economy grew by 6.32% in the first quater, a fact highlighted by Scott Bessent"
- This might be tokenized as, ('The',' economy',' grew',' by',' ','6','.','32','%',' in',' the',' first',' qu','ater',',',' a',' fact',' highlighted',' by',' Scott',' B','ess','ent')
- Tokenizers give common 'words' their own token, but less common 'words' (numbers, names, mispellings) are broken up into multiple tokens

Deep Learning for Text Analysis

Tokenization

- Byte Pair Encoding (BPE):
  - Start with a sentence “The dog ate the cat”

Deep Learning for Text Analysis

Tokenization

- Byte Pair Encoding (BPE):
  - Start with a sentence "The dog ate the cat"
  - Convert into lower case words, ('the','dog','ate','the','cat')

Deep Learning for Text Analysis

Tokenization

- Byte Pair Encoding (BPE):
  - Start with a sentence "The dog ate the cat"
  - Convert into lower case words, ('the','dog','ate','the','cat')
  - Base vocabulary is ('t','h','e','d','o','g','a','c')

Deep Learning for Text Analysis

Tokenization

- Byte Pair Encoding (BPE):
  - Start with a sentence "The dog ate the cat"
  - Convert into lower case words, ('the','dog','ate','the','cat')
  - Base vocabulary is ('t','h','e','d','o','g','a','c')
  - Consider the pairs found in the words ('th','he','do','og','at','te','th','he','ca','at')

Deep Learning for Text Analysis

Tokenization

- Byte Pair Encoding (BPE):
  - Start with a sentence "The dog ate the cat"
  - Convert into lower case words, ('the','dog','ate','the','cat')
  - Base vocabulary is ('t','h','e','d','o','g','a','c')
  - Consider the pairs found in the words ('th','he','do','og','at','te','th','he','ca','at')
  - Count frequencies: ('th':2,'he':2,'do','og','at':2,'te','ca')

Deep Learning for Text Analysis

Tokenization

- Byte Pair Encoding (BPE):
  - Start with a sentence "The dog ate the cat"
  - Convert into lower case words, ('the','dog','ate','the','cat')
  - Base vocabulary is ('t','h','e','d','o','g','a','c')
  - Consider the pairs found in the words ('th','he','do','og','at','te','th','he','ca','at')
  - Count frequencies: ('th':2,'he':2,'do','og','at':2,'te','ca')
  - The most common one is chosen to be added to the vocabulary—('t','h','e','d','o','g','a','c','th')

Deep Learning for Text Analysis

Tokenization

- Byte Pair Encoding (BPE):
  - Start with a sentence "The dog ate the cat"
  - Convert into lower case words, ('the','dog','ate','the','cat')
  - Base vocabulary is ('t','h','e','d','o','g','a','c')
  - Consider the pairs found in the words ('th','he','do','og','at','te','th','he','ca','at')
  - Count frequencies: ('th':2,'he':2,'do','og','at':2,'te','ca')
  - The most common one is chosen to be added to the vocabulary—('t','h','e','d','o','g','a','c','th')
  - Recount frequencies of pairs: ('the':2,'do','og','at':2,'te','ca',) and continue process until a given vocabulary size

Deep Learning for Text Analysis

Tokenization

- Byte Pair Encoding (BPE):
  - Start with a sentence "The dog ate the cat"
  - Convert into lower case words, ('the','dog','ate','the','cat')
  - Base vocabulary is ('t','h','e','d','o','g','a','c')
  - Consider the pairs found in the words ('th','he','do','og','at','te','th','he','ca','at')
  - Count frequencies: ('th':2,'he':2,'do','og','at':2,'te','ca')
  - The most common one is chosen to be added to the vocabulary—('t','h','e','d','o','g','a','c','th')
  - Recount frequencies of pairs: ('the':2,'do','og','at':2,'te','ca',) and continue process until a given vocabulary size
  - If we continued this process, we would add 'at', then 'ate' and 'cat', and then 'dog'

Deep Learning for Text Analysis

Tokenization

- Alternative common tokenizers:
  - Word Piece Encoding—very similar to byte-pair

Deep Learning for Text Analysis

Tokenization

- Alternative common tokenizers:
  - Word Piece Encoding—very similar to byte-pair
  - Sentence Piece Encoding—starts with a large vocabulary and tries to make it smaller (the reverse of Byte Pair and Word Piece encoding)—arguably less dependent on spacing, and can therefore be applied to languages like Japanese, Korean, etc.

Deep Learning for Text Analysis

Tokenization

- Alternative common tokenizers:
  - Word Piece Encoding—very similar to byte-pair
  - Sentence Piece Encoding—starts with a large vocabulary and tries to make it smaller (the reverse of Byte Pair and Word Piece encoding)—arguably less dependent on spacing, and can therefore be applied to languages like Japanese, Korean, etc.
- BERT used WordPiece tokenization with a 30,000 token vocabulary

Deep Learning for Text Analysis

Tokenization

- Alternative common tokenizers:
  - Word Piece Encoding—very similar to byte-pair
  - Sentence Piece Encoding—starts with a large vocabulary and tries to make it smaller (the reverse of Byte Pair and Word Piece encoding)—arguably less dependent on spacing, and can therefore be applied to languages like Japanese, Korean, etc.
- BERT used WordPiece tokenization with a 30,000 token vocabulary
- RoBERTa used a Byte-Pair Encoding tokenization with a 50,265 token vocabulary

Deep Learning for Text Analysis

RoBERTa Model Architecture

- BERT and RoBERTa used the same model architecture for the unsupervised step and the supervised step:

Deep Learning for Text Analysis

RoBERTa Model Architecture

- BERT and RoBERTa used the same model architecture for the unsupervised step and the supervised step:
  - BERT used 12 stacked transformer blocks, with 12 heads and 768-dimensional embeddings

Deep Learning for Text Analysis

RoBERTa Model Architecture

- BERT and RoBERTa used the same model architecture for the unsupervised step and the supervised step:
  - BERT used 12 stacked transformer blocks, with 12 heads and 768-dimensional embeddings
  - RoBERTa used 24 stacked transformer blocks, with 16 heads and 768-dimensional embeddings

Deep Learning for Text Analysis

Unsupervised Learning Step

- This step uses a very large amount of raw text

Deep Learning for Text Analysis

Unsupervised Learning Step

- This step uses a very large amount of raw text
- In principle, similar to word2vec and GloVe, but transformer architecture more directly models the relationship between words

Deep Learning for Text Analysis

Unsupervised Learning Step

- This step uses a very large amount of raw text
- In principle, similar to word2vec and GloVe, but transformer architecture more directly models the relationship between words
- BERT:
  - Unsupervised Task 1: 15% of words are masked and the model guesses the missing words

Deep Learning for Text Analysis

Unsupervised Learning Step

- This step uses a very large amount of raw text
- In principle, similar to word2vec and GloVe, but transformer architecture more directly models the relationship between words
- BERT:
  - Unsupervised Task 1: 15% of words are masked and the model guesses the missing words
  - Unsupervised Task 2: The model is given either two sentences that appeared following each other in the raw text or two sentences at random and the model guesses whether one sentence follows the other

Deep Learning for Text Analysis

Unsupervised Learning Step

- This step uses a very large amount of raw text
- In principle, similar to word2vec and GloVe, but transformer architecture more directly models the relationship between words
- BERT:
  - Unsupervised Task 1: 15% of words are masked and the model guesses the missing words
  - Unsupervised Task 2: The model is given either two sentences that appeared following each other in the raw text or two sentences at random and the model guesses whether one sentence follows the other
- RoBERTa only used unsupervised task 1

Deep Learning for Text Analysis

Supervised Learning Step

- In the supervised learning step, the model is fine-tuned

Deep Learning for Text Analysis

Supervised Learning Step

- In the supervised learning step, the model is fine-tuned
  - Fine-tuning refers to taking a previously trained model as a starting point and training all parameters for an additional number of epochs on different training data

Deep Learning for Text Analysis

Supervised Learning Step

- In the supervised learning step, the model is fine-tuned
  - Fine-tuning refers to taking a previously trained model as a starting point and training all parameters for an additional number of epochs on different training data
  - Typically, the original data set is larger and trained for a greater number of epochs while the fine-tuning happens on a smaller data set for fewer epochs

Deep Learning for Text Analysis

Supervised Learning Step

- In the supervised learning step, the model is fine-tuned
  - Fine-tuning refers to taking a previously trained model as a starting point and training all parameters for an additional number of epochs on different training data
  - Typically, the original data set is larger and trained for a greater number of epochs while the fine-tuning happens on a smaller data set for fewer epochs
  - In the case of BERT and RoBERTa, the supervised step uses a number of tasks

Deep Learning for Text Analysis

Supervised Learning Step

- BERT and RoBERTa use the following tasks in the supervised learning step:
  - Sentence pair classification, e.g. pairs of sentences are labeled being as logically compatible, incompatible, or neutral (Williams, Nangia and Bowman, 2018)

Deep Learning for Text Analysis

Supervised Learning Step

- BERT and RoBERTa use the following tasks in the supervised learning step:
  - Sentence pair classification, e.g. pairs of sentences are labeled being as logically compatible, incompatible, or neutral (Williams, Nangia and Bowman, 2018)
  - Single sentence classification, e.g. sentiment (Socher et al., 2013)

Deep Learning for Text Analysis

Supervised Learning Step

- BERT and RoBERTa use the following tasks in the supervised learning step:
  - Sentence pair classification, e.g. pairs of sentences are labeled being as logically compatible, incompatible, or neutral (Williams, Nangia and Bowman, 2018)
  - Single sentence classification, e.g. sentiment (Socher et al., 2013)
  - Question answering, e.g. crowdsourced worked read a wikipedia article, generate a question, and code which words in the article answer the question (Rajpurkar et al., 2016)

Deep Learning for Text Analysis

Supervised Learning Step

- BERT and RoBERTa use the following tasks in the supervised learning step:
  - Sentence pair classification, e.g. pairs of sentences are labeled being as logically compatible, incompatible, or neutral (Williams, Nangia and Bowman, 2018)
  - Single sentence classification, e.g. sentiment (Socher et al., 2013)
  - Question answering, e.g. crowdsourced worked read a wikipedia article, generate a question, and code which words in the article answer the question (Rajpurkar et al., 2016)
  - Single sentence tagging, e.g. tag the named entities in a sentences (Tjong Kim Sang and De Meulder, 2003)

Deep Learning for Text Analysis

Supervised Learning Step

- BERT and RoBERTa use the following tasks in the supervised learning step:
  - Sentence pair classification, e.g. pairs of sentences are labeled being as logically compatible, incompatible, or neutral (Williams, Nangia and Bowman, 2018)
  - Single sentence classification, e.g. sentiment (Socher et al., 2013)
  - Question answering, e.g. crowdsourced worked read a wikipedia article, generate a question, and code which words in the article answer the question (Rajpurkar et al., 2016)
  - Single sentence tagging, e.g. tag the named entities in a sentences (Tjong Kim Sang and De Meulder, 2003)
- BERT and RoBERTa used many labeled previously published datasets

Deep Learning for Text Analysis

Supervised Learning Step

- BERT and RoBERTa use the following tasks in the supervised learning step:
  - Sentence pair classification, e.g. pairs of sentences are labeled being as logically compatible, incompatible, or neutral (Williams, Nangia and Bowman, 2018)
  - Single sentence classification, e.g. sentiment (Socher et al., 2013)
  - Question answering, e.g. crowdsourced worked read a wikipedia article, generate a question, and code which words in the article answer the question (Rajpurkar et al., 2016)
  - Single sentence tagging, e.g. tag the named entities in a sentences (Tjong Kim Sang and De Meulder, 2003)
- BERT and RoBERTa used many labeled previously published datasets
- The size of the labeled dataset (in total) is large, but substantially smaller than the 160GB of text fed to the model in the unsupervised step

Deep Learning for Text Analysis

Supervised Learning Step

- BERT and RoBERTa can do the 12 things they were trained to do (including sentiment analysis)—if you are not trying to do one of those 12 things, you might have to:
  - Get lucky because someone else has fine-tuned BERT/RoBERTa on a labeled dataset which matches your goals

Deep Learning for Text Analysis

Supervised Learning Step

- BERT and RoBERTa can do the 12 things they were trained to do (including sentiment analysis)—if you are not trying to do one of those 12 things, you might have to:
  - Get lucky because someone else has fine-tuned BERT/RoBERTa on a labeled dataset which matches your goals
  - Collect a labeled dataset and fine-tune BERT/RoBERTa on that dataset (more computationally costly)

Deep Learning for Text Analysis

Supervised Learning Step

- BERT and RoBERTa can do the 12 things they were trained to do (including sentiment analysis)—if you are not trying to do one of those 12 things, you might have to:
  - Get lucky because someone else has fine-tuned BERT/RoBERTa on a labeled dataset which matches your goals
  - Collect a labeled dataset and fine-tune BERT/RoBERTa on that dataset (more computationally costly)
  - Collect a labeled dataset and use the final layer [CLS] tokens as an embedding, which represents the model's knowledge of an entire sentence (less computationally costly)

Deep Learning for Text Analysis

- Application 1: Sentiment in IMDB Reviews

	Mode	Per. Mode	Y=0	Y=1	All
LASSO, BOW:					
Train	0	50.1	88.1	89.8	89.0
Test	1	50.1	85.4	86.6	86.0
NN, BOW, 0 Hidden Layers:					
Train	0	50.1	87.2	90.6	88.9
Test	1	50.1	85.1	87.8	86.5
NN, BOW, 1 Hidden Layer:					
Train	0	50.1	99.8	99.8	99.8
Test	1	50.1	86.5	84.3	85.4
NN, BOW, 2 Hidden Layers:					
Train	0	50.1	99.9	99.4	99.6
Test	1	50.1	87.0	81.6	84.3
LSTM:					
Train	0	50.1	94.3	86.0	90.2
Test	1	50.1	90.7	81.2	85.9
Transformer:					
Train	0	50.1	93.6	88.5	91.0
Test	0	50.1	90.2	83.6	86.9
RoBERTa (pre-trained):					
Train	0	52.6	96.6	97.0	96.8
Test	1	50.0	94.8	95.7	95.2

Deep Learning for Text Analysis

- Application 1: Sentiment in IMDB Reviews

	Mode	Per. Mode	Y=0	Y=1	All
LASSO, BOW:					
Train	0	50.1	88.1	89.8	89.0
Test	1	50.1	85.4	86.6	86.0
NN, BOW, 0 Hidden Layers:					
Train	0	50.1	87.2	90.6	88.9
Test	1	50.1	85.1	87.8	86.5
NN, BOW, 1 Hidden Layer:					
Train	0	50.1	99.8	99.8	99.8
Test	1	50.1	86.5	84.3	85.4
NN, BOW, 2 Hidden Layers:					
Train	0	50.1	99.9	99.4	99.6
Test	1	50.1	87.0	81.6	84.3
LSTM:					
Train	0	50.1	94.3	86.0	90.2
Test	1	50.1	90.7	81.2	85.9
Transformer:					
Train	0	50.1	93.6	88.5	91.0
Test	0	50.1	90.2	83.6	86.9
RoBERTa (pre-trained):					
Train	0	52.6	96.6	97.0	96.8
Test	1	50.0	94.8	95.7	95.2

- The pre-trained model is the best fitting model (though I can't be sure that RoBERTa was not trained on the IMBD dataset)

Deep Learning for Text Analysis

- Application 2: Sentiment in Rotten Tomatoes Reviews

	Mode	Per. Mode	Y=0	Y=1	All
Naive Bayes, BOW:					
Train	1	69.6	66.8	79.4	75.5
Test	1	68.4	62.9	79.3	74.2
LASSO, BOW:					
Train	1	69.6	77.4	94.7	89.5
Test	1	68.4	60.4	84.8	77.1
NN, BOW, 0 Hidden Layers:					
Train	1	69.6	72.1	94.0	87.4
Test	1	68.4	59.4	88.1	79.1
NN, BOW, 1 Hidden Layer:					
Train	1	69.6	95.9	97.9	97.3
Test	1	68.4	60.0	86.9	78.4
NN, BOW, 2 Hidden Layers:					
Train	1	69.6	91.7	99.5	97.1
Test	1	68.4	56.8	88.2	78.3
LSTM:					
Train	1	69.6	70.3	92.4	85.7
Test	1	68.4	56.3	88.6	78.4
Transformer:					
Train	1	69.6	78.4	91.2	87.3
Test	1	68.4	64.8	84.6	78.3
RoBERTa (pre-trained):					
Train	1	69.6	59.6	89.0	80.1
Test	1	68.4	60.1	89.0	79.9

Deep Learning for Text Analysis

- Application 2: Sentiment in Rotten Tomatoes Reviews

	Mode	Per. Mode	Y=0	Y=1	All
Naive Bayes, BOW:					
Train	1	69.6	66.8	79.4	75.5
Test	1	68.4	62.9	79.3	74.2
LASSO, BOW:					
Train	1	69.6	77.4	94.7	89.5
Test	1	68.4	60.4	84.8	77.1
NN, BOW, 0 Hidden Layers:					
Train	1	69.6	72.1	94.0	87.4
Test	1	68.4	59.4	88.1	79.1
NN, BOW, 1 Hidden Layer:					
Train	1	69.6	95.9	97.9	97.3
Test	1	68.4	60.0	86.9	78.4
NN, BOW, 2 Hidden Layers:					
Train	1	69.6	91.7	99.5	97.1
Test	1	68.4	56.8	88.2	78.3
LSTM:					
Train	1	69.6	70.3	92.4	85.7
Test	1	68.4	56.3	88.6	78.4
Transformer:					
Train	1	69.6	78.4	91.2	87.3
Test	1	68.4	64.8	84.6	78.3
RoBERTa (pre-trained):					
Train	1	69.6	59.6	89.0	80.1
Test	1	68.4	60.1	89.0	79.9

- The pre-trained model is the best fitting model (but not by much)

Deep Learning for Text Analysis

Pretrained Models

- As they are bigger models, pre-trained models can take longer to run than smaller models (even though you are not training the pre-trained models)

Deep Learning for Text Analysis

Pretrained Models

- As they are bigger models, pre-trained models can take longer to run than smaller models (even though you are not training the pre-trained models)
- Application 3 requires an additional step, fine-tuning, which is even more computationally costly

Deep Learning for Text Analysis

- Application 3: Coding whether newspaper articles are discussing the economy

	Mode	Per. Mode	Y=0	Y=1	All
Naive Bayes, BOW:					
Train	1	58.6	78.1	85.9	82.7
Test	1	58.6	72.1	76.2	74.5
LASSO, BOW:					
Train	1	58.6	89.9	94.3	92.5
Test	1	58.6	64.6	72.0	68.9
LASSO, CBOW:					
Train	1	58.6	67.4	84.2	77.3
Test	1	58.6	71.5	79.6	76.3
NN, CBOW, 0 Hidden Layers:					
Train	1	58.6	62.0	82.6	74.1
Test	1	58.6	66.2	80.6	74.6
NN, CBOW, 1 Hidden Layer:					
Train	1	58.6	68.4	81.8	76.3
Test	1	58.6	71.5	79.4	76.1
NN, CBOW, 2 Hidden Layers:					
Train	1	58.6	66.8	86.1	78.1
Test	1	58.6	69.5	81.3	76.4
LSTM, Pre. Emb.:					
Train	1	58.6	72.0	81.4	77.5
Test	1	58.6	74.4	76.0	75.3
Transformer, Pre. Emb.:					
Train	1	58.6	44.9	94.9	74.2
Test	1	58.6	46.6	93.8	74.2
Pre-trained, Fine-tuned:					
Train					71.0
Test					67.8

Deep Learning for Text Analysis

Pretrained Models

- As they are bigger models, pre-trained models can take longer to run than smaller models (even though you are not training the pre-trained models)
- Application 3 requires an additional step, fine-tuning, which even more computationally costly
- The performance of fine-tuned RoBERTa was not very good in Application 3

Deep Learning for Text Analysis

Pretrained Models

- As they are bigger models, pre-trained models can take longer to run than smaller models (even though you are not training the pre-trained models)
- Application 3 requires an additional step, fine-tuning, which even more computationally costly
- The performance of fine-tuned RoBERTa was not very good in Application 3
- Can we avoid fine-tuning?

Deep Learning for Text Analysis

Pretrained Models

- As they are bigger models, pre-trained models can take longer to run than smaller models (even though you are not training the pre-trained models)
- Application 3 requires an additional step, fine-tuning, which even more computationally costly
- The performance of fine-tuned RoBERTa was not very good in Application 3
- Can we avoid fine-tuning?
  - Potential answer: Large Language Models (LLMs)

Deep Learning for Text Analysis

Using LLMs for Coding Text

- Large language models refers to models with an extremely large number of parameters

Deep Learning for Text Analysis

Using LLMs for Coding Text

- Large language models refers to models with an extremely large number of parameters
- Some models (like RoBERTa) that are far too expensive to estimate on a home computer are referred to as small language models in the literature

Deep Learning for Text Analysis

Using LLMs for Coding Text

- Large language models refers to models with an extremely large number of parameters
- Some models (like RoBERTa) that are far too expensive to estimate on a home computer are referred to as small language models in the literature
- For expositional clarity, I will use the terms as follows:
  - Small Language Model (SLM): a model that you can train on your home computer (e.g. LASSO, LSTM, etc.)

Deep Learning for Text Analysis

Using LLMs for Coding Text

- Large language models refers to models with an extremely large number of parameters
- Some models (like RoBERTa) that are far too expensive to estimate on a home computer are referred to as small language models in the literature
- For expositional clarity, I will use the terms as follows:
  - Small Language Model (SLM): a model that you can train on your home computer (e.g. LASSO, LSTM, etc.)
  - Medium Language Model (MLM): a model that is pre-trained, but for which fine-tuning is feasible (but potentially costly) on a home computer

Deep Learning for Text Analysis

Using LLMs for Coding Text

- Large language models refers to models with an extremely large number of parameters
- Some models (like RoBERTa) that are far too expensive to estimate on a home computer are referred to as small language models in the literature
- For expositional clarity, I will use the terms as follows:
  - Small Language Model (SLM): a model that you can train on your home computer (e.g. LASSO, LSTM, etc.)
  - Medium Language Model (MLM): a model that is pre-trained, but for which fine-tuning is feasible (but potentially costly) on a home computer
  - Large Language Model: a model that can be given chat-type input and is not feasible to train or fine-tune on a home computer (i.e. Llama, DeepSeek, ChatGPT)

Deep Learning for Text Analysis

Using LLMs for Coding Text

- Steps in LLM model:
  - Tokenization of text
  - Unsupervised training
  - Supervised fine-tuning
  - Prompt injection

Deep Learning for Text Analysis

Using LLMs for Coding Text

- Steps in LLM model:
  - Tokenization of text
  - Unsupervised training
  - Supervised fine-tuning
  - Prompt injection
- Tokenization is very similar to what we discussed for BERT and RoBERTa, but LLMs may use a larger vocabulary

Deep Learning for Text Analysis

Using LLMs for Coding Text

- Steps in LLM model:
  - Tokenization of text
  - Unsupervised training
  - Supervised fine-tuning
  - Prompt injection
- Tokenization is very similar to what we discussed for BERT and RoBERTa, but LLMs may use a larger vocabulary
- Unsupervised step is very similar to what we discussed, but dataset might be bigger

Deep Learning for Text Analysis

Using LLMs for Coding Text

- Steps in LLM model:
  - Tokenization of text
  - Unsupervised training
  - Supervised fine-tuning
  - Prompt injection
- Tokenization is very similar to what we discussed for BERT and RoBERTa, but LLMs may use a larger vocabulary
- Unsupervised step is very similar to what we discussed, but dataset might be bigger
- Supervised fine-tuning is a little different

Deep Learning for Text Analysis

Using LLMs for Coding Text

- Steps in LLM model:
  - Tokenization of text
  - Unsupervised training
  - Supervised fine-tuning
  - Prompt injection
- Tokenization is very similar to what we discussed for BERT and RoBERTa, but LLMs may use a larger vocabulary
- Unsupervised step is very similar to what we discussed, but dataset might be bigger
- Supervised fine-tuning is a little different
- Prompt injection is unique to commercial models with a chat-type interface

Deep Learning for Text Analysis

Using LLMs for Coding Text

- Basic architecture:
  - Most LLMs rely on Generative Pre-trained Transformers (GPTs) as the basic architecture

Deep Learning for Text Analysis

Using LLMs for Coding Text

- Basic architecture:
  - Most LLMs rely on Generative Pre-trained Transformers (GPTs) as the basic architecture
  - A GPT is in many ways similar to word2Vec and BERT/RoBERTa—it fills in missing words in a document

Deep Learning for Text Analysis

Using LLMs for Coding Text

- Basic architecture:
  - Most LLMs rely on Generative Pre-trained Transformers (GPTs) as the basic architecture
  - A GPT is in many ways similar to word2Vec and BERT/RoBERTa—it fills in missing words in a document
  - A GPT works in order, so that the next word is filled in using previous words (as opposed to word2Vec and BERT/RoBERTa, which look in both directions), as the main goal is to predict the next word in a response

Deep Learning for Text Analysis

Using LLMs for Coding Text

- Basic architecture:
  - Most LLMs rely on Generative Pre-trained Transformers (GPTs) as the basic architecture
  - A GPT is in many ways similar to word2Vec and BERT/RoBERTa—it fills in missing words in a document
  - A GPT works in order, so that the next word is filled in using previous words (as opposed to word2Vec and BERT/RoBERTa, which look in both directions), as the main goal is to predict the next word in a response
  - While word2vec relied on averaging latent embeddings, the model GPTs use to predict the next token is more complex—they use a transformer model to predict the next token

Deep Learning for Text Analysis

Using LLMs for Coding Text

- Unsupervised Learning Step for LLMs:
  - LLMs typically use even more data (as much as a factor of 10 larger)

Deep Learning for Text Analysis

Using LLMs for Coding Text

- Unsupervised Learning Step for LLMs:
  - LLMs typically use even more data (as much as a factor of 10 larger)
  - Training data could consist of the Common Crawl (filtered to eliminate undesireable training data), millions of scanned OCR'd books, and all of wikipedia

Deep Learning for Text Analysis

Using LLMs for Coding Text

- Supervised fine-tuning of LLMs:
  - Varies from provider to provider and is proprietary knowledge

Deep Learning for Text Analysis

Using LLMs for Coding Text

- Supervised fine-tuning of LLMs:
  - Varies from provider to provider and is proprietary knowledge
  - Tends not to rely on standard published datasets like BERT and RoBERTa did

Deep Learning for Text Analysis

Using LLMs for Coding Text

- Supervised fine-tuning of LLMs:
  - Varies from provider to provider and is proprietary knowledge
  - Tends not to rely on standard published datasets like BERT and RoBERTa did
  - Instead, tends to rely on in-house datasets where coders write prompts and answers or grade answers to promts

Deep Learning for Text Analysis

Using LLMs for Coding Text

- Supervised fine-tuning of LLMs:
  - Varies from provider to provider and is proprietary knowledge
  - Tends not to rely on standard published datasets like BERT and RoBERTa did
  - Instead, tends to rely on in-house datasets where coders write prompts and answers or grade answers to promts
  - In some cases, the 'coder' grading prompts may be users of the product

Deep Learning for Text Analysis

Using LLMs for Coding Text

- System Prompt step:
  - System prompts are added to all queries asked of a user through a commercial LLM interface

Deep Learning for Text Analysis

Using LLMs for Coding Text

- System Prompt step:
  - System prompts are added to all queries asked of a user through a commercial LLM interface
  - Using the model directly could bypass system prompts in some cases—for example using an open source model

Deep Learning for Text Analysis

Using LLMs for Coding Text

- System Prompt step:
  - System prompts are added to all queries asked of a user through a commercial LLM interface
  - Using the model directly could bypass system prompts in some cases—for example using an open source model
  - Some examples from Microsoft copilots system prompt: "You also know you don't have human experiences, so you NEVER make statements or claims which insinuate or imply you are or wish to be conscious, sentient, alive, or human, or speculate about one day evolving to be.", "DO NOT engage in stereotyping, including negative stereotyping of majority groups. If asked controversial topics, provide careful thoughts and objective information without downplaying harmful content or implying there are reasonable perspectives on both sides."

Deep Learning for Text Analysis

Using LLMs for Coding Text

- System Prompt step:
  - Goals are to make the chatbot more useful and protect the business interests of the company making the chatbot—i.e. chatbot shouldn't tell users to do illegal things to prevent legal exposure for the chatbot

Deep Learning for Text Analysis

Using LLMs for Coding Text

- System Prompt step:
  - Goals are to make the chatbot more useful and protect the business interests of the company making the chatbot—i.e. chatbot shouldn't tell users to do illegal things to prevent legal exposure for the chatbot
  - Some of this could be achieved in the supervised step—the training set could label a response on which bridge is the tallest as a bad response if the human coder decided this was a bad things to include because it might be asked by suicidal individuals, but the system prompts can be used to get the missing examples.

Deep Learning for Text Analysis

Using LLMs for Coding Text

- Summary:
  - Users poses a prompt to a chatbot

Deep Learning for Text Analysis

Using LLMs for Coding Text

- Summary:
  - Users poses a prompt to a chatbot
  - A system prompt is added to the user prompt

Deep Learning for Text Analysis

Using LLMs for Coding Text

- Summary:
  - Users poses a prompt to a chatbot
  - A system prompt is added to the user prompt
  - The overall prompt is tokenized using an algorithm like Byte Pair encoding

Deep Learning for Text Analysis

Using LLMs for Coding Text

- Summary:
  - Users poses a prompt to a chatbot
  - A system prompt is added to the user prompt
  - The overall prompt is tokenized using an algorithm like Byte Pair encoding
  - A GPT is used to predict the next token multiple times, until an [END RESPONSE] token is generated

Deep Learning for Text Analysis

Using LLMs for Coding Text

- Summary:
  - Users poses a prompt to a chatbot
  - A system prompt is added to the user prompt
  - The overall prompt is tokenized using an algorithm like Byte Pair encoding
  - A GPT is used to predict the next token multiple times, until an [END RESPONSE] token is generated
    - The GPT model is a transformer model that was trained using an unsupervised step and a fined-tuning supervised step

Deep Learning for Text Analysis

Using LLMs for Coding Text

- Summary:
  - Users poses a prompt to a chatbot
  - A system prompt is added to the user prompt
  - The overall prompt is tokenized using an algorithm like Byte Pair encoding
  - A GPT is used to predict the next token multiple times, until an [END RESPONSE] token is generated
    - The GPT model is a transformer model that was trained using an unsupervised step and a fined-tuning supervised step
  - The tokens are converted into a response, using the reverse of the tokenization algorithm

Deep Learning for Text Analysis

Using LLMs for Coding Text

- LLMs can in theory be used to code sentiment, topic, or ideology, like other text analysis techniques

Deep Learning for Text Analysis

Using LLMs for Coding Text

- LLMs can in theory be used to code sentiment, topic, or ideology, like other text analysis techniques
- Example: "Please respond on a scale of 1 to 2, with 1 indicating negative sentiment and 2 indicating positive sentiment. How would you describe the sentiment of the following movie review on the 1 to 2 scale (please confine your answer to just the integers 1 and 2). Please ensure the 1 to 2 rating appears first in your response. Here is the movie review:"

Deep Learning for Text Analysis

Using LLMs for Coding Text

- LLMs can in theory be used to code sentiment, topic, or ideology, like other text analysis techniques
- Example: "Please respond on a scale of 1 to 2, with 1 indicating negative sentiment and 2 indicating positive sentiment. How would you describe the sentiment of the following movie review on the 1 to 2 scale (please confine your answer to just the integers 1 and 2). Please ensure the 1 to 2 rating appears first in your response. Here is the movie review:"
- An advantage of LLMs is that they don't require labeled training data, which can be very expensive to collect

Deep Learning for Text Analysis

Using LLMs for Coding Text

- LLMs can in theory be used to code sentiment, topic, or ideology, like other text analysis techniques
- Example: "Please respond on a scale of 1 to 2, with 1 indicating negative sentiment and 2 indicating positive sentiment. How would you describe the sentiment of the following movie review on the 1 to 2 scale (please confine your answer to just the integers 1 and 2). Please ensure the 1 to 2 rating appears first in your response. Here is the movie review:"
- An advantage of LLMs is that they don't require labeled training data, which can be very expensive to collect
- A disadvantage is that one may not know if the LLM will provide good response if you don't have training data to compare against

Deep Learning for Text Analysis

Using LLMs for Coding Text

- A second disadvantage is that the LLM may not respond in the format you request

Deep Learning for Text Analysis

Using LLMs for Coding Text

- A second disadvantage is that the LLM may not respond in the format you request
- "The sentiment of the article about the economy is: \n\n**2**\n\n The article reports on the contradictory consumption indicators in Luxembourg..."

Deep Learning for Text Analysis

Using LLMs for Coding Text

- A second disadvantage is that the LLM may not respond in the format you request
- "The sentiment of the article about the economy is: \n\n**2**\n\n The article reports on the contradictory consumption indicators in Luxembourg..."
- Worse examples might not include a numeric coding, or might include an invalid numeric coding (e.g. 2.6)

Deep Learning for Text Analysis

Using LLMs for Coding Text

- Examples below used the llama3-8b-8192 model from Groq

Deep Learning for Text Analysis

Using LLMs for Coding Text

- Examples below used the llama3-8b-8192 model from Groq
- Groq is a provider of AI computing services and allows users to cheaply run open models like Llama and DeepSeek, which would be very slow on a home computer

Deep Learning for Text Analysis

Using LLMs for Coding Text

- Examples below used the llama3-8b-8192 model from Groq
- Groq is a provider of AI computing services and allows users to cheaply run open models like Llama and DeepSeek, which would be very slow on a home computer
- Note that Groq $\neq$ Grok (despite being pronounced the same)—Grok is an LLM developed by twitter

Deep Learning for Text Analysis

Using LLMs for Coding Text

- Examples below used the llama3-8b-8192 model from Groq
- Groq is a provider of AI computing services and allows users to cheaply run open models like Llama and DeepSeek, which would be very slow on a home computer
- Note that Groq $\neq$ Grok (despite being pronounced the same)—Grok is an LLM developed by twitter
- llama3-8b-8192 has 8 billion parameters and an 8192 token context window (i.e. the number of tokens that can be supplied in a prompt)

Deep Learning for Text Analysis

Using LLMs for Coding Text

- Examples below used the llama3-8b-8192 model from Groq
- Groq is a provider of AI computing services and allows users to cheaply run open models like Llama and DeepSeek, which would be very slow on a home computer
- Note that Groq $\neq$ Grok (despite being pronounced the same)—Grok is an LLM developed by twitter
- llama3-8b-8192 has 8 billion parameters and an 8192 token context window (i.e. the number of tokens that can be supplied in a prompt)
- Facebook developed the Llama model, but Groq fine-tunes the model using in house coders

Deep Learning for Text Analysis

- Application 1: Sentiment in IMDB Reviews

	Mode	Per. Mode	Y=0	Y=1	All
LASSO, BOW:					
Train	0	50.1	88.1	89.8	89.0
Test	1	50.1	85.4	86.6	86.0
LSTM:					
Train	0	50.1	94.3	86.0	90.2
Test	1	50.1	90.7	81.2	85.9
Transformer:					
Train	0	50.1	93.6	88.5	91.0
Test	0	50.1	90.2	83.6	86.9
RoBERTa (pre-trained):					
Train	0	52.6	96.6	97.0	96.8
Test	1	50.0	94.8	95.7	95.2
LLM:					
Train					
Test					92.2

Deep Learning for Text Analysis

- Application 1: Sentiment in IMDB Reviews

	Mode	Per. Mode	Y=0	Y=1	All
LASSO, BOW:					
Train	0	50.1	88.1	89.8	89.0
Test	1	50.1	85.4	86.6	86.0
LSTM:					
Train	0	50.1	94.3	86.0	90.2
Test	1	50.1	90.7	81.2	85.9
Transformer:					
Train	0	50.1	93.6	88.5	91.0
Test	0	50.1	90.2	83.6	86.9
RoBERTa (pre-trained):					
Train	0	52.6	96.6	97.0	96.8
Test	1	50.0	94.8	95.7	95.2
LLM:					
Train					
Test					92.2

- The LLM has good performance, beatings the SLMs, but underperforms the pre-trained (MLM) model

Deep Learning for Text Analysis

- Application 1: Sentiment in IMDB Reviews

	Mode	Per. Mode	Y=0	Y=1	All
LASSO, BOW:					
Train	0	50.1	88.1	89.8	89.0
Test	1	50.1	85.4	86.6	86.0
LSTM:					
Train	0	50.1	94.3	86.0	90.2
Test	1	50.1	90.7	81.2	85.9
Transformer:					
Train	0	50.1	93.6	88.5	91.0
Test	0	50.1	90.2	83.6	86.9
RoBERTa (pre-trained):					
Train	0	52.6	96.6	97.0	96.8
Test	1	50.0	94.8	95.7	95.2
LLM:					
Train					
Test					92.2

- The LLM has good performance, beatings the SLMs, but underperforms the pre-trained (MLM) model
- LLM has the disadvantage that it will cost money, but the advantage that it won't take forever to run

Deep Learning for Text Analysis

- Application 2: Sentiment in Rotten Tomatoes Reviews

	Mode	Per. Mode	Y=0	Y=1	All
LASSO, BOW:					
Train	1	69.6	77.4	94.7	89.5
Test	1	68.4	60.4	84.8	77.1
NN, BOW, 0 Hidden Layers:					
Train	1	69.6	72.1	94.0	87.4
Test	1	68.4	59.4	88.1	79.1
LSTM:					
Train	1	69.6	70.3	92.4	85.7
Test	1	68.4	56.3	88.6	78.4
Transformer:					
Train	1	69.6	78.4	91.2	87.3
Test	1	68.4	64.8	84.6	78.3
RoBERTa (pre-trained):					
Train	1	69.6	59.6	89.0	80.1
Test	1	68.4	60.1	89.0	79.9
LLM:					
Train	1	69.6	87.6	84.8	85.7
Test	1	68.4	86.3	83.8	84.6

Deep Learning for Text Analysis

- Application 2: Sentiment in Rotten Tomatoes Reviews

	Mode	Per. Mode	Y=0	Y=1	All
LASSO, BOW:					
Train	1	69.6	77.4	94.7	89.5
Test	1	68.4	60.4	84.8	77.1
NN, BOW, 0 Hidden Layers:					
Train	1	69.6	72.1	94.0	87.4
Test	1	68.4	59.4	88.1	79.1
LSTM:					
Train	1	69.6	70.3	92.4	85.7
Test	1	68.4	56.3	88.6	78.4
Transformer:					
Train	1	69.6	78.4	91.2	87.3
Test	1	68.4	64.8	84.6	78.3
RoBERTa (pre-trained):					
Train	1	69.6	59.6	89.0	80.1
Test	1	68.4	60.1	89.0	79.9
LLM:					
Train	1	69.6	87.6	84.8	85.7
Test	1	68.4	86.3	83.8	84.6

- The LLM is the best performing model

Deep Learning for Text Analysis

Using LLMs for Coding Text

- LLMs worked pretty well for sentiment

Deep Learning for Text Analysis

Using LLMs for Coding Text

- LLMs worked pretty well for sentiment
- MLMs didn't work that well for topic—can LLMs perform better?

Deep Learning for Text Analysis

Using LLMs for Coding Text

- LLMs worked pretty well for sentiment
- MLMs didn't work that well for topic—can LLMs perform better?
- The results below coded whether the topic of an article was about the economy in three different ways:
  - Topic and sentiment in the same prompt (a 0-5 scale, with 0 indicating that the article is not about the economy)

Deep Learning for Text Analysis

Using LLMs for Coding Text

- LLMs worked pretty well for sentiment
- MLMs didn't work that well for topic—can LLMs perform better?
- The results below coded whether the topic of an article was about the economy in three different ways:
  - Topic and sentiment in the same prompt (a 0-5 scale, with 0 indicating that the article is not about the economy)
  - Binary scale for whether the article was about the economy

Deep Learning for Text Analysis

Using LLMs for Coding Text

- LLMs worked pretty well for sentiment
- MLMs didn't work that well for topic—can LLMs perform better?
- The results below coded whether the topic of an article was about the economy in three different ways:
  - Topic and sentiment in the same prompt (a 0-5 scale, with 0 indicating that the article is not about the economy)
  - Binary scale for whether the article was about the economy
  - Confidence that the article is discussing the economy on a 0-100 percent certainty scale, with a logit model used to determine a cutoff

Deep Learning for Text Analysis

- Application 3: Coding whether newspaper articles are discussing the economy

	Mode	Per. Mode	Uncoded	Y=0	Y=1	All
Naive Bayes, BOW:						
Train	1	58.6		78.1	85.9	82.7
Test	1	58.6		72.1	76.2	74.5
LASSO, CBOW:						
Train	1	58.6		67.4	84.2	77.3
Test	1	58.6		71.5	79.6	76.3
NN, CBOW, 2 Hidden Layers:						
Train	1	58.6		66.8	86.1	78.1
Test	1	58.6		69.5	81.3	76.4
LSTM, Pre. Emb.:						
Train	1	58.6		72.0	81.4	77.5
Test	1	58.6		74.4	76.0	75.3
Transformer, Pre. Emb.:						
Train	1	58.6		44.9	94.9	74.2
Test	1	58.6		46.6	93.8	74.2
Pre-trained, Fine-tuned:						
Train						71.0
Test						67.8
LLM, Comb. Coding						
Train	1	56.7	0.3	25.6	96.6	65.8
Test	1	59.2	0.4	24.8	97.6	67.9
LLM, Binary Scale						
Train	1	56.7	0.3	56.0	74.6	66.5
Test	1	59.2	0.4	55.4	69.6	63.8
LLM, 0-100 Scale						
Train	1	56.7	0.3	14.4	92.1	58.4
Test	1	59.2	0.2	16.3	95.2	63.0

Deep Learning for Text Analysis

Using LLMs for Coding Text

- Application 3: Coding whether newspaper articles are discussing the economy
  - LASSO with GloVe embedding was correct 76.3% of the time

Deep Learning for Text Analysis

Using LLMs for Coding Text

- Application 3: Coding whether newspaper articles are discussing the economy
  - LASSO with GloVe embedding was correct 76.3% of the time
  - The best supervised learning model was a neural network with 2 hidden layers and GloVe embeddings, with 76.4% correct

Deep Learning for Text Analysis

Using LLMs for Coding Text

- Application 3: Coding whether newspaper articles are discussing the economy
  - LASSO with GloVe embedding was correct 76.3% of the time
  - The best supervised learning model was a neural network with 2 hidden layers and GloVe embeddings, with 76.4% correct
  - Fine-tuned RoBERTa model was correct 67.8% of the time

Deep Learning for Text Analysis

Using LLMs for Coding Text

- Application 3: Coding whether newspaper articles are discussing the economy
  - LASSO with GloVe embedding was correct 76.3% of the time
  - The best supervised learning model was a neural network with 2 hidden layers and GloVe embeddings, with 76.4% correct
  - Fine-tuned RoBERTa model was correct 67.8% of the time
  - An LLM (llama3-8b-8192) was correct 67.9%, 63.8%, and 63.0% of the time, depending on the prompt

Deep Learning for Text Analysis

Using LLMs for Coding Text

- Application 3: Coding whether newspaper articles are discussing the economy
  - LASSO with GloVe embedding was correct 76.3% of the time
  - The best supervised learning model was a neural network with 2 hidden layers and GloVe embeddings, with 76.4% correct
  - Fine-tuned RoBERTa model was correct 67.8% of the time
  - An LLM (llama3-8b-8192) was correct 67.9%, 63.8%, and 63.0% of the time, depending on the prompt
  - The LLM could be viewed as having poor performance as it under-performs LASSO with GloVe embeddings, but it does skip to expensive step of labeling examples

Deep Learning for Text Analysis

Using LLMs for Coding Text

- Additional approaches
  - Use an LLM with fine tuning:
    - Not feasible on home computer as model has too many parameters

Deep Learning for Text Analysis

Using LLMs for Coding Text

- Additional approaches
  - Use an LLM with fine tuning:
    - Not feasible on home computer as model has too many parameters
    - Partially defeats the point of LLMs, as the point was to avoid having to collect labeled data

Deep Learning for Text Analysis

Using LLMs for Coding Text

- Additional approaches
  - Use an LLM with fine tuning:
    - Not feasible on home computer as model has too many parameters
    - Partially defeats the point of LLMs, as the point was to avoid having to collect labeled data
    - Groq allows you to submit data for fine-tuning

Deep Learning for Text Analysis

Using LLMs for Coding Text

- Additional approaches
  - Use an LLM with fine tuning:
    - Not feasible on home computer as model has too many parameters
    - Partially defeats the point of LLMs, as the point was to avoid having to collect labeled data
    - Groq allows you to submit data for fine-tuning
  - Use the top layers of an LLM as inputs to supervised learning model—Parameter-Efficient Fine-Tuning (PEFT)

Deep Learning for Text Analysis

Using LLMs for Coding Text

- Additional approaches
  - Use an LLM with fine tuning:
    - Not feasible on home computer as model has too many parameters
    - Partially defeats the point of LLMs, as the point was to avoid having to collect labeled data
    - Groq allows you to submit data for fine-tuning
  - Use the top layers of an LLM as inputs to supervised learning model—Parameter-Efficient Fine-Tuning (PEFT)
    - Again, partially defeats the point of LLMs

Deep Learning for Text Analysis

Some Other Applications

- Measuring Ideology:
  - Mens and Gallego (2025) measure ideology in tweets by asking LLMs to code the tweet directly. They claim the LLMs perform well and beat a fine-tuned BERT model

Deep Learning for Text Analysis

Some Other Applications

- Measuring Ideology:
  - Mens and Gallego (2025) measure ideology in tweets by asking LLMs to code the tweet directly. They claim the LLMs perform well and beat a fine-tuned BERT model
  - Erhard et al. (2025) code leftwing and rightwing populism in parliamentary speeches using a transformer model. They claim good results

Deep Learning for Text Analysis

Some Other Applications

- Measuring Ideology:
  - Mens and Gallego (2025) measure ideology in tweets by asking LLMs to code the tweet directly. They claim the LLMs perform well and beat a fine-tuned BERT model
  - Erhard et al. (2025) code leftwing and rightwing populism in parliamentary speeches using a transformer model. They claim good results
  - Lai et al. (2024) code the ideology of youtube videos using a fine-tuned BERT model, which predicts the ideology of the video based on the metadata

Deep Learning for Text Analysis

Some Other Applications

- Measuring Ideology:
  - Mens and Gallego (2025) measure ideology in tweets by asking LLMs to code the tweet directly. They claim the LLMs perform well and beat a fine-tuned BERT model
  - Erhard et al. (2025) code leftwing and rightwing populism in parliamentary speeches using a transformer model. They claim good results
  - Lai et al. (2024) code the ideology of youtube videos using a fine-tuned BERT model, which predicts the ideology of the video based on the metadata
- Measuring Topic:
  - Daniel J. Hopkins (2025) measure the prevalence of identity-based news coverage using a BERT model

Deep Learning for Text Analysis

Some Other Applications

- Measuring Ideology:
  - Mens and Gallego (2025) measure ideology in tweets by asking LLMs to code the tweet directly. They claim the LLMs perform well and beat a fine-tuned BERT model
  - Erhard et al. (2025) code leftwing and rightwing populism in parliamentary speeches using a transformer model. They claim good results
  - Lai et al. (2024) code the ideology of youtube videos using a fine-tuned BERT model, which predicts the ideology of the video based on the metadata
- Measuring Topic:
  - Daniel J. Hopkins (2025) measure the prevalence of identity-based news coverage using a BERT model
  - Vishwanath (2025) measures the prevalence of race-based symbolic representation in the communication of members of Congress using the LASSO, random forests, and support vector machines

Deep Learning for Text Analysis

Some Other Applications

- Misc. applications:
  - Silva, Pullan and Wäckerle (2025) measure whether members of parliament uses feminine or masculine speech using logistic ridge-regression and decision trees

Deep Learning for Text Analysis

Some Other Applications

- Misc. applications:
  - Silva, Pullan and Wäckerle (2025) measure whether members of parliament uses feminine or masculine speech using logistic ridge-regression and decision trees
- Some patterns:
  - LLMs still seem to be used less, MLMs are catching on, and SLM are still used

Deep Learning for Text Analysis

Some Other Applications

- Misc. applications:
  - Silva, Pullan and Wäckerle (2025) measure whether members of parliament uses feminine or masculine speech using logistic ridge-regression and decision trees
- Some patterns:
  - LLMs still seem to be used less, MLMs are catching on, and SLM are still used
  - Smaller models are associated with deeper applications and LLMs are associated with shallower applications

Deep Learning for Text Analysis

Some Other Applications

- Misc. applications:
  - Silva, Pullan and Wäckerle (2025) measure whether members of parliament uses feminine or masculine speech using logistic ridge-regression and decision trees
- Some patterns:
  - LLMs still seem to be used less, MLMs are catching on, and SLM are still used
  - Smaller models are associated with deeper applications and LLMs are associated with shallower applications
  - Most papers develop measures, with Lai et al. (2024) being an exception, proxying for the content of the video with the metadata

Deep Learning for Text Analysis

Some Other Applications

- Misc. applications:
  - Silva, Pullan and Wäckerle (2025) measure whether members of parliament uses feminine or masculine speech using logistic ridge-regression and decision trees
- Some patterns:
  - LLMs still seem to be used less, MLMs are catching on, and SLM are still used
  - Smaller models are associated with deeper applications and LLMs are associated with shallower applications
  - Most papers develop measures, with Lai et al. (2024) being an exception, proxying for the content of the video with the metadata
  - Most papers using human-coded training data, but Lai et al. (2024) and Silva, Pullan and Wäckerle (2025) are exceptions

Deep Learning for Text Analysis

Conclusions

- Dollar Cost:
  - Factor 1: Collecting Labeled Data—fancier models need more labeled data to achieve good performance

Deep Learning for Text Analysis

Conclusions

- Dollar Cost:
  - Factor 1: Collecting Labeled Data—fancier models need more labeled data to achieve good performance
    - In Kayser and Peress (2021), we spent about 4000 dollars collecting a relatively small amount of training data (about 160 hours of work from trained RAs)

Deep Learning for Text Analysis

Conclusions

- Dollar Cost:
  - Factor 1: Collecting Labeled Data—fancier models need more labeled data to achieve good performance
    - In Kayser and Peress (2021), we spent about 4000 dollars collecting a relatively small amount of training data (about 160 hours of work from trained RAs)
    - Bird (2019) spent about 40 hours collecting a minimal training data set

Deep Learning for Text Analysis

Conclusions

- Dollar Cost:
  - Factor 1: Collecting Labeled Data—fancier models need more labeled data to achieve good performance
    - In Kayser and Peress (2021), we spent about 4000 dollars collecting a relatively small amount of training data (about 160 hours of work from trained RAs)
    - Bird (2019) spent about 40 hours collecting a minimal training data set
  - Factor 2: Paying for APIs—large datasets lead to more cost and 'better' models lead to more cost

Deep Learning for Text Analysis

Conclusions

- Dollar Cost:
  - Factor 1: Collecting Labeled Data—fancier models need more labeled data to achieve good performance
    - In Kayser and Peress (2021), we spent about 4000 dollars collecting a relatively small amount of training data (about 160 hours of work from trained RAs)
    - Bird (2019) spent about 40 hours collecting a minimal training data set
  - Factor 2: Paying for APIs—large datasets lead to more cost and 'better' models lead to more cost
    - In Kayser and Peress (2021), we analyzed 3 million newspaper articles, with about 2.8 billion LLM tokens—the cheapest models on Groq (the cheapest provider) would cost about $1,500, a somewhat better DeepSeek AI model would cost $25,000, and the state of the art OpenAI model would cost $70,000

Deep Learning for Text Analysis

Conclusions

- Time Cost:
  - In Kayser and Peress (2021), dictionaries, naive Bayes, LASSO, and support vector machines required a few hours to complete

Deep Learning for Text Analysis

Conclusions

- Time Cost:
  - In Kayser and Peress (2021), dictionaries, naive Bayes, LASSO, and support vector machines required a few hours to complete
  - Pre-trained BERT would require 1-2 days to process sentiment alone

Deep Learning for Text Analysis

Conclusions

- Time Cost:
  - In Kayser and Peress (2021), dictionaries, naive Bayes, LASSO, and support vector machines required a few hours to complete
  - Pre-trained BERT would require 1-2 days to process sentiment alone
  - Fine-tuned BERT would require 300 days for just one measure (not feasible)

Deep Learning for Text Analysis

Conclusions

- Time Cost:
  - In Kayser and Peress (2021), dictionaries, naive Bayes, LASSO, and support vector machines required a few hours to complete
  - Pre-trained BERT would require 1-2 days to process sentiment alone
  - Fine-tuned BERT would require 300 days for just one measure (not feasible)
  - Smallest LLM would require about 100 days to complete, DeepSeek model would require about 400 days to complete (basically not feasible without tricks to get around rate limits)

Deep Learning for Text Analysis

Conclusions

- Time Cost:
  - In Kayser and Peress (2021), dictionaries, naive Bayes, LASSO, and support vector machines required a few hours to complete
  - Pre-trained BERT would require 1-2 days to process sentiment alone
  - Fine-tuned BERT would require 300 days for just one measure (not feasible)
  - Smallest LLM would require about 100 days to complete, DeepSeek model would require about 400 days to complete (basically not feasible without tricks to get around rate limits)
  - Bird's (2019) data was about one-third the size—rather than waste effort coding things very efficiently, he relied on the Stony Brook computer cluster, and computation time was less than a day

Deep Learning for Text Analysis

Conclusions

- Application Depth:
  - A minimal application may report a time series while a deeper application will use text to generate a measure used as a dependent variable and/or and independent variable

Deep Learning for Text Analysis

Conclusions

- Application Depth:
  - A minimal application may report a time series while a deeper application will use text to generate a measure used as a dependent variable and/or and independent variable
  - If text is used to generate a dependent variable (Bird, 2019; Kayser and Peress, 2021), measurement error is less of an issue and sampling can be used to minimize computation cost and time

Deep Learning for Text Analysis

Conclusions

- Application Depth:
  - A minimal application may report a time series while a deeper application will use text to generate a measure used as a dependent variable and/or and independent variable
  - If text is used to generate a dependent variable (Bird, 2019; Kayser and Peress, 2021), measurement error is less of an issue and sampling can be used to minimize computation cost and time
  - If text is used to generate an independent variable that is not aggregated (Daniel J. Hopkins, 2025), sampling can be used to minimize computational cost and time

Deep Learning for Text Analysis

Conclusions

- Application Depth:
  - A minimal application may report a time series while a deeper application will use text to generate a measure used as a dependent variable and/or and independent variable
  - If text is used to generate a dependent variable (Bird, 2019; Kayser and Peress, 2021), measurement error is less of an issue and sampling can be used to minimize computation cost and time
  - If text is used to generate an independent variable that is not aggregated (Daniel J. Hopkins, 2025), sampling can be used to minimize computational cost and time
  - If text is used to generate an aggregated independent variable (Kayser and Peress, Forthcoming; Silva, Pullan and Wäckerle, 2025; Vishwanath, 2025), sampling can introduce massive measurement error, partially explaining the negative correlation between application depth and model size observed in the literature

Deep Learning for Text Analysis

Conclusions

- Sociological Factors:
  - Even if MLMs and LLMs could be used to skip the collection of labeled data, readers may expect a test on labeled data

Deep Learning for Text Analysis

Conclusions

- Sociological Factors:
  - Even if MLMs and LLMs could be used to skip the collection of labeled data, readers may expect a test on labeled data
  - 'Fancier' supervised learning models do not necessarily lead to large improvement over less fancy ones, but readers may expect them

Deep Learning for Text Analysis

Conclusions

- Sociological Factors:
  - Evaluation is often based on ability to predict labeled data, but the final goal is to make correct inferences on theoretical relationships

Deep Learning for Text Analysis

Conclusions

- Sociological Factors:
  - Evaluation is often based on ability to predict labeled data, but the final goal is to make correct inferences on theoretical relationships
    - Example 1: LLMs could be argued to have performed poorly on determining whether an article was discussing the economy, but the LLMs may simply be using a different threshold than the human coders, which may not manifest as an incorrect inference on a theoretical relationship (but we'll never know!!)

Deep Learning for Text Analysis

Conclusions

- Sociological Factors:
  - Evaluation is often based on ability to predict labeled data, but the final goal is to make correct inferences on theoretical relationships
    - Example 1: LLMs could be argued to have performed poorly on determining whether an article was discussing the economy, but the LLMs may simply be using a different threshold than the human coders, which may not manifest as an incorrect inference on a theoretical relationship (but we'll never know!!)
    - Example 2: Method A may have a poorer fit on a test set, but the errors may be largely uncorrelated, leading to less error when aggregated. Method B may have a better fit on a test set, but the error may be correlated leading to more error when aggregated (but we'll never know!!)

Deep Learning for Text Analysis

Conclusions

- The Rabbit Hole Problem:

Deep Learning for Text Analysis

Conclusions

- The Rabbit Hole Problem:
  - If I estimate a LASSO with bag of words on a labeled dataset, I can easily test whether the LASSO worked

Deep Learning for Text Analysis

Conclusions

- The Rabbit Hole Problem:
  - If I estimate a LASSO with bag of words on a labeled dataset, I can easily test whether the LASSO worked
  - If I fine-tune RoBERTa on a labeled dataset, maybe I didn't used the right number of epochs, or the latest fanciest version of a pre-trained modal. Or maybe I should have used parameter-efficient fine-tuning

Deep Learning for Text Analysis

Conclusions

- The Rabbit Hole Problem:
  - If I estimate a LASSO with bag of words on a labeled dataset, I can easily test whether the LASSO worked
  - If I fine-tune RoBERTa on a labeled dataset, maybe I didn't used the right number of epochs, or the latest fanciest version of a pre-trained modal. Or maybe I should have used parameter-efficient fine-tuning
  - If I use ChatGPT on my text data and it performs poorly, maybe I should have used the obviously superior Claude model. Or maybe I didn't use the best prompt

Deep Learning for Text Analysis

Conclusions

- The Rabbit Hole Problem:
  - If I estimate a LASSO with bag of words on a labeled dataset, I can easily test whether the LASSO worked
  - If I fine-tune RoBERTa on a labeled dataset, maybe I didn't used the right number of epochs, or the latest fanciest version of a pre-trained modal. Or maybe I should have used parameter-efficient fine-tuning
  - If I use ChatGPT on my text data and it performs poorly, maybe I should have used the obviously superior Claude model. Or maybe I didn't use the best prompt
  - The observed literature is full of fancier models that happened to work out on minimal applications, making it seem that these methods are more universally successful than they are

Deep Learning for Text Analysis

Conclusions

- The Rabbit Hole Problem:
  - If I estimate a LASSO with bag of words on a labeled dataset, I can easily test whether the LASSO worked
  - If I fine-tune RoBERTa on a labeled dataset, maybe I didn't used the right number of epochs, or the latest fanciest version of a pre-trained modal. Or maybe I should have used parameter-efficient fine-tuning
  - If I use ChatGPT on my text data and it performs poorly, maybe I should have used the obviously superior Claude model. Or maybe I didn't use the best prompt
  - The observed literature is full of fancier models that happened to work out on minimal applications, making it seem that these methods are more universally successful than they are
  - Degrees of freedom generate unfair opportunities for criticism, which can lead to extensive defensive work

Deep Learning for Text Analysis

Conclusions

- Fancier models can achieve better performance in certain circumstances, but temper expectations so you don't fall into a rabbit hole

Deep Learning for Text Analysis

Conclusions

- Fancier models can achieve better performance in certain circumstances, but temper expectations so you don't fall into a rabbit hole
- Choose a strategy—fancy model with minimal application or workhorse model with deep application

Deep Learning for Text Analysis

Conclusions

- Fancier models can achieve better performance in certain circumstances, but temper expectations so you don't fall into a rabbit hole
- Choose a strategy—fancy model with minimal application or workhorse model with deep application
  - A fancy model can insulate you from criticism that your application is minimal

Deep Learning for Text Analysis

Conclusions

- Fancier models can achieve better performance in certain circumstances, but temper expectations so you don't fall into a rabbit hole
- Choose a strategy—fancy model with minimal application or workhorse model with deep application
  - A fancy model can insulate you from criticism that your application is minimal
  - A deep application can insulate you from criticism that your model isn't fancy

Deep Learning for Text Analysis

Conclusions

- Fancier models can achieve better performance in certain circumstances, but temper expectations so you don't fall into a rabbit hole
- Choose a strategy—fancy model with minimal application or workhorse model with deep application
  - A fancy model can insulate you from criticism that your application is minimal
  - A deep application can insulate you from criticism that your model isn't fancy
- In most cases, reviewers will expect a labeled dataset, which will be costly (in time and money) to collect, but if you can quickly run an LLM without excessive time and money investment, it might be worth it to risk it and try to publish a paper without a labeled dataset

Deep Learning for Text Analysis

Additional Implementation Considerations

- Quick Startup: Use Google Colab with Tensorflow:

Deep Learning for Text Analysis

Additional Implementation Considerations

- Quick Startup: Use Google Colab with Tensorflow:
  - Tensorflow is generally easier to learn (if you use it through Keras)

Deep Learning for Text Analysis

Additional Implementation Considerations

- Quick Startup: Use Google Colab with Tensorflow:
  - Tensorflow is generally easier to learn (if you use it through Keras)
  - LLMs can run much faster if you employ an NVidia GPU

Deep Learning for Text Analysis

Additional Implementation Considerations

- Quick Startup: Use Google Colab with Tensorflow:
  - Tensorflow is generally easier to learn (if you use it through Keras)
  - LLMs can run much faster if you employ an NVidia GPU
  - You can set up Google Colab to use a GPU

Deep Learning for Text Analysis

Additional Implementation Considerations

- More Involved: Run python on your home machine

Deep Learning for Text Analysis

Additional Implementation Considerations

- More Involved: Run python on your home machine
  - Tensorflow can in theory run on a GPU, but you need a particular (older) version of Python running Linux (so likely not practical on home machine)

Deep Learning for Text Analysis

Additional Implementation Considerations

- More Involved: Run python on your home machine
  - Tensorflow can in theory run on a GPU, but you need a particular (older) version of Python running Linux (so likely not practical on home machine)
  - Pytorch has a larger entry cost, but can easily use a GPU if you have a decent one (e.g. NVidia RTX 4060)

Deep Learning for Text Analysis

Additional Implementation Considerations

- More Involved: Run python on your home machine
  - Tensorflow can in theory run on a GPU, but you need a particular (older) version of Python running Linux (so likely not practical on home machine)
  - Pytorch has a larger entry cost, but can easily use a GPU if you have a decent one (e.g. NVidia RTX 4060)
  - Easiest approach:
    - Install Python 3.12

Deep Learning for Text Analysis

Additional Implementation Considerations

- More Involved: Run python on your home machine
  - Tensorflow can in theory run on a GPU, but you need a particular (older) version of Python running Linux (so likely not practical on home machine)
  - Pytorch has a larger entry cost, but can easily use a GPU if you have a decent one (e.g. NVidia RTX 4060)
  - Easiest approach:
    - Install Python 3.12
    - Install Visual Studio Code

Deep Learning for Text Analysis

Additional Implementation Considerations

- More Involved: Run python on your home machine
  - Tensorflow can in theory run on a GPU, but you need a particular (older) version of Python running Linux (so likely not practical on home machine)
  - Pytorch has a larger entry cost, but can easily use a GPU if you have a decent one (e.g. NVidia RTX 4060)
  - Easiest approach:
    - Install Python 3.12
    - Install Visual Studio Code
    - Install Python extension in Visual Studio Code

Deep Learning for Text Analysis

Additional Implementation Considerations

- More Involved: Run python on your home machine
  - Tensorflow can in theory run on a GPU, but you need a particular (older) version of Python running Linux (so likely not practical on home machine)
  - Pytorch has a larger entry cost, but can easily use a GPU if you have a decent one (e.g. NVidia RTX 4060)
  - Easiest approach:
    - Install Python 3.12
    - Install Visual Studio Code
    - Install Python extension in Visual Studio Code
    - Install Jupyter Notebook extension in Visual Studio Code

Deep Learning for Text Analysis

Additional Implementation Considerations

- More Involved: Run python on your home machine
  - Tensorflow can in theory run on a GPU, but you need a particular (older) version of Python running Linux (so likely not practical on home machine)
  - Pytorch has a larger entry cost, but can easily use a GPU if you have a decent one (e.g. NVidia RTX 4060)
  - Easiest approach:
    - Install Python 3.12
    - Install Visual Studio Code
    - Install Python extension in Visual Studio Code
    - Install Jupyter Notebook extension in Visual Studio Code
    - Create a Jupyter Notebook

Deep Learning for Text Analysis

Additional Implementation Considerations

- More Involved: Run python on your home machine
  - Tensorflow can in theory run on a GPU, but you need a particular (older) version of Python running Linux (so likely not practical on home machine)
  - Pytorch has a larger entry cost, but can easily use a GPU if you have a decent one (e.g. NVidia RTX 4060)
  - Easiest approach:
    - Install Python 3.12
    - Install Visual Studio Code
    - Install Python extension in Visual Studio Code
    - Install Jupyter Notebook extension in Visual Studio Code
    - Create a Jupyter Notebook
    - Install python libraries using "% pip install ABC"

Deep Learning for Text Analysis

Additional Implementation Considerations

- More Involved: Run python on your home machine
  - Tensorflow can in theory run on a GPU, but you need a particular (older) version of Python running Linux (so likely not practical on home machine)
  - Pytorch has a larger entry cost, but can easily use a GPU if you have a decent one (e.g. NVidia RTX 4060)
  - Easiest approach:
    - Install Python 3.12
    - Install Visual Studio Code
    - Install Python extension in Visual Studio Code
    - Install Jupyter Notebook extension in Visual Studio Code
    - Create a Jupyter Notebook
    - Install python libraries using "% pip install ABC"
    - Build your model in Pytorch

References I

Barbera, Pablo. 2015. "Birds of the Same Feather Tweet Together: Bayesian Ideal Point Estimation Using Twitter Data." *Political Analysis* 23:76–91.

Bird, Robert. 2019. "The Media as Ideological 'Gatekeepers'? Findings of Minimal Partisan Selection Bias in Television News Coverage." Working Paper.

Daniel J. Hopkins, Yphtach Lelkes, Samuel Wolken. 2025. "The Rise of and Demand for Identity-oriented Media Coverage." *American Journal of Political Science* 69:483—500.

References II

Devlin, Jacob, Ming-Wei Chang, Kenton Lee and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Vol. 1 Minneapolis, Minnesota: Association for Computational Linguistics pp. 4171–4186.
URL: *https://aclanthology.org/N19-1423/*

Erhard, Lukas, Sara Hanke, Uwe Remer, Agnieszka Falenska and Raphael Heiko Heiberger. 2025. "PopBERT. Detecting Populism and Its Host Ideologies in the German Bundestag." *Political Analysis* 33:1—17.

Griewank, A. and A. Walther. 2008. *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*. Society for Industrial and Applied Mathematics.

References III

Kayser, Mark and Michael Peress. 2021. "Does the Media Cover the Economy Accurately? An Analysis of Sixteen Developed Democracies." *Quarterly Journal of Political Science* 16:1–33.

Kayser, Mark and Michael Peress. Forthcoming. "Do Voters Respond to the Economy or to News Reporting on the Economy?" *Journal of Politics* .

Lai, Angela, Megan A. Brown, James Bisbee, Joshua A. Tucker, Jonathan Nagler and Richard Bonneau. 2024. "Estimating the Ideology of Political YouTube Videos." *Political Analysis* 32:345—360.

Liu, Yinhan, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer and Veselin Stoyanov. 2019. "RoBERTa: A Robustly Optimized BERT Pretraining Approach.".
URL: *https://arxiv.org/abs/1907.11692*

References IV

McFadden, Daniel. 1978. Modelling the Choice of Residential Location. In *Spatial Interaction Theory and Planning Models*, ed. A. Karlquist et al. Amsterdam: North-Holland pp. 75–96.

Mens, Gael Le and Aina Gallego. 2025. "Positioning Political Texts with Large Language Models by Asking and Averaging." *Political Analysis* 33:274—282.

Mikolov, Tomas, Ilya Sutskever, Kai Chen, Greg Corrado and Jeffrey Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *Proceedings of the Conference and Workshop on Neural Information Processing Systems (NIPS'13)*.

Pennington, Jeffrey, Richard Socher and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics pp. 1532–1543.

References V

Rajpurkar, Pranav, Jian Zhang, Konstantin Lopyrev and Percy Liang. 2016. SQuAD: 100,000+ Questions for Machine Comprehension of Text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Austin, Texas: Association for Computational Linguistics pp. 2383–2392. **URL:** *https://aclanthology.org/D16-1264/*

Silva, Bruno Castanho, Danielle Pullan and Jens Wäckerle. 2025. "Blending in or Standing Out? Gendered Political Communication in 24 Democracies." *American Journal of Political Science* 69:653—668.

References VI

Socher, Richard, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng and Christopher Potts. 2013. Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Seattle, Washington, USA: Association for Computational Linguistics pp. 1631–1642.
URL: *https://aclanthology.org/D13-1170/*

Tjong Kim Sang, Erik F. and Fien De Meulder. 2003. Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*. Edmonton, Canada: Association for Computational Linguistics pp. 142–147.
URL: *https://aclanthology.org/W03-0419/*

References VII

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser and Illia Polosukhin. 2017. Attention Is All You Need. In *Advances in Neural Information Processing Systems*. Vol. 30.
URL: *https://arxiv.org/abs/1706.03762*

Vishwanath, Arjun. 2025. "Race, Legislative Speech, and Symbolic Representation in Congress." *American Journal of Political Science* 69:578—593.

Williams, Adina, Nikita Nangia and Samuel R. Bowman. 2018. A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. New Orleans, Louisiana: Association for Computational Linguistics pp. 1112–1122.
URL: *https://aclanthology.org/N18-1101/*