Sequencing Legal DNA

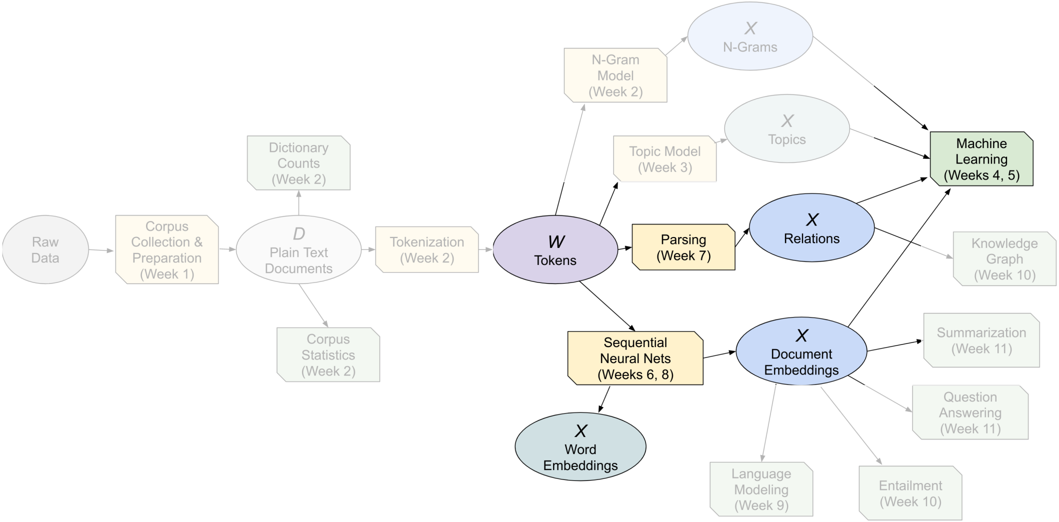
NLP for Law and Political Economy

8. Document Embeddings

Q&A Page

bit.ly/NLP-QA08

Course Progress (Weeks 5-8)



Outline

What is (Document) Embedding?

“Embedding”: a lower-dimensional dense vector representation of a higher-dimensional object

- also refers to algorithm for making such vectors

What is (Document) Embedding?

"Embedding": a lower-dimensional dense vector representation of a higher-dimensional object

- also refers to algorithm for making such vectors

Document vectors:

- quantitative analysis of language requires that documents be transformed to numbers – that is, vectors.

What is (Document) Embedding?

"Embedding": a lower-dimensional dense vector representation of a higher-dimensional object

- also refers to algorithm for making such vectors

Document vectors:

- quantitative analysis of language requires that documents be transformed to numbers – that is, vectors.
- standard approach:
  - represent documents as sparse vectors of token counts/frequencies.
  - e.g., to compute document similarity measures, to learn topics, or to use as features in supervised learning

What is (Document) Embedding?

"Embedding": a lower-dimensional dense vector representation of a higher-dimensional object

- also refers to algorithm for making such vectors

Document vectors:

- quantitative analysis of language requires that documents be transformed to numbers – that is, vectors.
- standard approach:
  - represent documents as sparse vectors of token counts/frequencies.
  - e.g., to compute document similarity measures, to learn topics, or to use as features in supervised learning
- **Embedding approach**:
  - low-dimensional dense vectors rather than high-dimensional sparse vectors
  - Embedding without neural nets:
    - PCA reductions of the document-term matrix
    - LDA topic shares

What is (Document) Embedding?

"Embedding": a lower-dimensional dense vector representation of a higher-dimensional object

- also refers to algorithm for making such vectors

Document vectors:

- quantitative analysis of language requires that documents be transformed to numbers – that is, vectors.
- standard approach:
  - represent documents as sparse vectors of token counts/frequencies.
  - e.g., to compute document similarity measures, to learn topics, or to use as features in supervised learning
- **Embedding approach**:
  - low-dimensional dense vectors rather than high-dimensional sparse vectors
  - Embedding without neural nets:
    - PCA reductions of the document-term matrix
    - LDA topic shares
  - Embedding with neural nets (today):
    - many useful ways to do this.

Embedding layers can produce document embeddings

- **Embedding layers** take a categorical variable as input and produce a low-dimensional dense representation.

Embedding layers can produce document embeddings

- **Embedding layers** take a categorical variable as input and produce a low-dimensional dense representation.

Can be used to produce document embeddings:

- Tokenize document to fixed length n_L
- Inputs are each word position, input categorical (word) to n_E-dimensional embedding layer:

$$\boldsymbol{x}_{1:n_L} = [\ \boldsymbol{x}_1 \quad \ldots \quad \boldsymbol{x}_t \quad \ldots \quad \boldsymbol{x}_{n_L}\ ]$$

- pipe to further hidden layers of network.

Embedding layers can produce document embeddings

- **Embedding layers** take a categorical variable as input and produce a low-dimensional dense representation.

Can be used to produce document embeddings:

- Tokenize document to fixed length n_L
- Inputs are each word position, input categorical (word) to n_E-dimensional embedding layer:

$$\boldsymbol{x}_{1:n_L} = [\ \boldsymbol{x}_1 \quad \ldots \quad \boldsymbol{x}_t \quad \ldots \quad \boldsymbol{x}_{n_L}\ ]$$

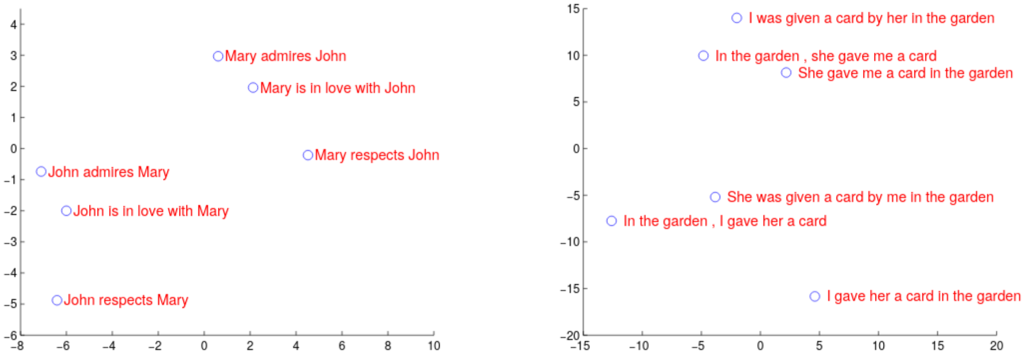
- pipe to further hidden layers of network.
- document embedding $=$ $n_L n_E$-dimensional vector of concatenated word embeddings.
  - computationally demanding and only works with short documents.

Autoencoder Encodings

- Autoencoder compresses a document (e.g. a sentence) into a vector to be reconstructed.
  - Can use the compressed representation as a document embedding.
- Standard (that is, non-transformer) autoencoder embeddings don't tend to work well for sentence similarity tasks because autoencoders try to reproduce the specific wording (reconstruction objective), rather than the semantic meaning.
  - transformer-based autoencoders, e.g. BART, address this issue (Week 9)

RNN's (e.g. Machine Translation) Produce Document Embeddings

- In Week 5, we saw that RNN machine translators produce a sentence vector that must be decoded into another language.
- if the vector produces a good translation, it must contain the important information in the sentence.



Outline

Word Vectors can produce Document Vectors

$$\vec{D} = \sum_{w \in D} a_w \vec{w}$$

- The "continuous bag of words" representation for document D is the sum, or the average (potentially weighted by a_w), of the vectors $\vec{w}$ for each word w in the document.
  - word vectors $\vec{w}$ constructed using Word2Vec or GloVe (pre-trained or trained on the corpus).
  - "Document" could be sentence, paragraph, section, etc.

Word Vectors can produce Document Vectors

$$\vec{D} = \sum_{w \in D} a_w \vec{w}$$

- The "continuous bag of words" representation for document D is the sum, or the average (potentially weighted by a_w), of the vectors $\vec{w}$ for each word w in the document.
  - word vectors $\vec{w}$ constructed using Word2Vec or GloVe (pre-trained or trained on the corpus).
  - "Document" could be sentence, paragraph, section, etc.
- Arora, Liang, and Ma (2017) provide a "tough to beat baseline", the SIF-weighted ("smoothed inverse frequency") average of the vectors:

$$a_w = \frac{\alpha}{\alpha + p_w}$$

where p_w is the probability (frequency) of the word and $\alpha = .001$ is a smoothing parameter.

Gennaro and Ash (2020): Emotions in Congress

- Corpus: floor speeches in U.S. Congress (House and Senate), 1858-2014
- Tokenization: stemmed nouns, adjectives, and verbs.

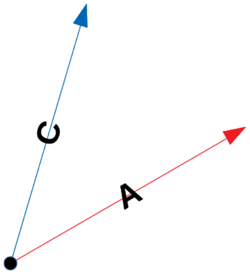
Gennaro and Ash (2020): Emotions in Congress

- Corpus: floor speeches in U.S. Congress (House and Senate), 1858-2014
- Tokenization: stemmed nouns, adjectives, and verbs.
- Measuring emotionality:
  - use LIWC dictionaries for "cognitive" (reason) and "affective" (emotion)
  - train Word2Vec on speeches (300 dims, eight-word context window)

Gennaro and Ash (2020): Emotions in Congress

- Corpus: floor speeches in U.S. Congress (House and Senate), 1858-2014
- Tokenization: stemmed nouns, adjectives, and verbs.
- Measuring emotionality:
  - use LIWC dictionaries for "cognitive" (reason) and "affective" (emotion)
  - train Word2Vec on speeches (300 dims, eight-word context window)
  - For each of the lexicons (cognitive and affective), form the centroid (average) vector:

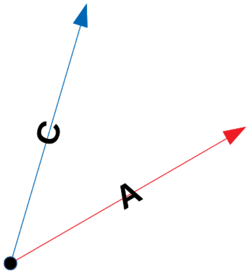
$\vec{A}$ = affective, $\vec{C}$ = cognitive centroid



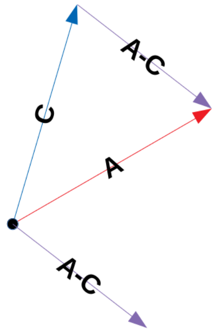
Gennaro and Ash (2020): Emotions in Congress

- Corpus: floor speeches in U.S. Congress (House and Senate), 1858-2014
- Tokenization: stemmed nouns, adjectives, and verbs.
- Measuring emotionality:
  - use LIWC dictionaries for "cognitive" (reason) and "affective" (emotion)
  - train Word2Vec on speeches (300 dims, eight-word context window)
  - For each of the lexicons (cognitive and affective), form the centroid (average) vector:

$\vec{A}$ = affective, $\vec{C}$ = cognitive centroid



$\vec{A} - \vec{C}$ = emotion-cognition dimension

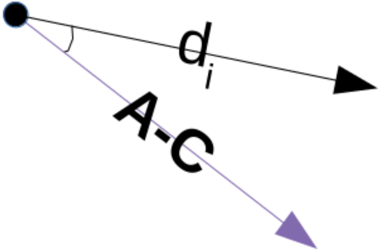


Emotionality Metric

- Construct document vector for speech i as the average of the word vectors in the speech (Arora, Liang, and Ma 2016)

Emotionality Metric

- Construct document vector for speech i as the average of the word vectors in the speech (Arora, Liang, and Ma 2016)

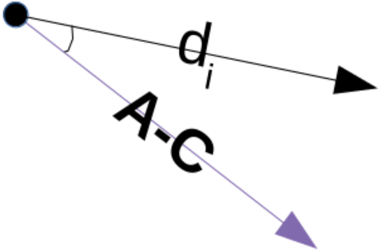


- Relative emotionality of i is **cosine similarity** to the emotion-cognition dimension:

$$Y_i = \frac{\vec{d}_i \cdot (\vec{A} - \vec{C})}{\left\|\vec{d}_i\right\| \left\|\vec{A} - \vec{C}\right\|}$$

Emotionality Metric

- Construct document vector for speech i as the average of the word vectors in the speech (Arora, Liang, and Ma 2016)

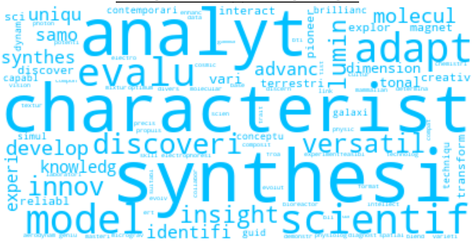


- Relative emotionality of i is **cosine similarity** to the emotion-cognition dimension:

$$Y_i = \frac{\vec{d}_i \cdot (\vec{A} - \vec{C})}{\left\| \vec{d}_i \right\| \left\| \vec{A} - \vec{C} \right\|}$$

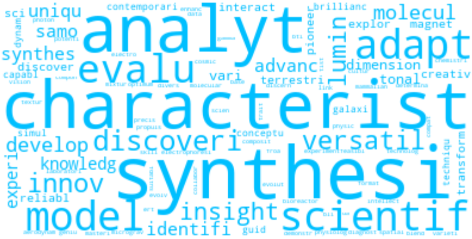
Increase in $Y_i \leftrightarrow$ shift towards emotion pole and away from cognition pole.

<u>Cognition Language</u>



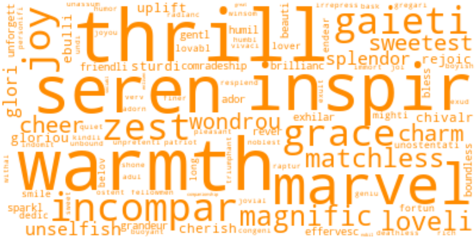
- "In my judgment, neither is true in the case of this amendment."
- "Is that correct?"
- "R. 15 contains a provision that is similar but, in fact, broader in scope."

Cognition Language



- "In my judgment, neither is true in the case of this amendment."
- "Is that correct?"
- "R. 15 contains a provision that is similar but, in fact, broader in scope."

Emotion Language



- “With joy in his heart and a smile on his face he graced practically every social occasion with a song.”
- “We Democrats may disagree, but we love our fellow men and we never hate them.”

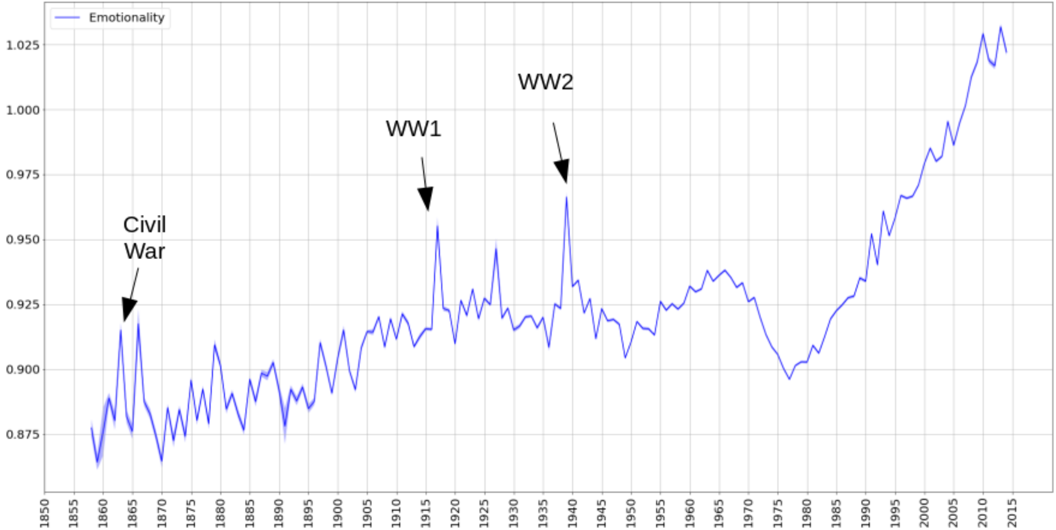
Human Validation

Table 3: HUMAN VALIDATION

	Full Sample			Restricted Sample English Comprehension			Restricted Sample Consistent Coding		
	(1) Accuracy	(2) Blank	(3) Sample	(4) Accuracy	(5) Blank	(6) Sample	(7) Accuracy	(8) Blank	(9) Sample
Panel A: Main Analysis									
Overall	**0.874**	**0.035**	**1714**	**0.923**	**0.029**	**1158**	**0.927**	**0.013**	**1388**

- the embedding measure matches human judgment much more often than a dictionary based measure.

Emotion Language in Congress, 1958-2014

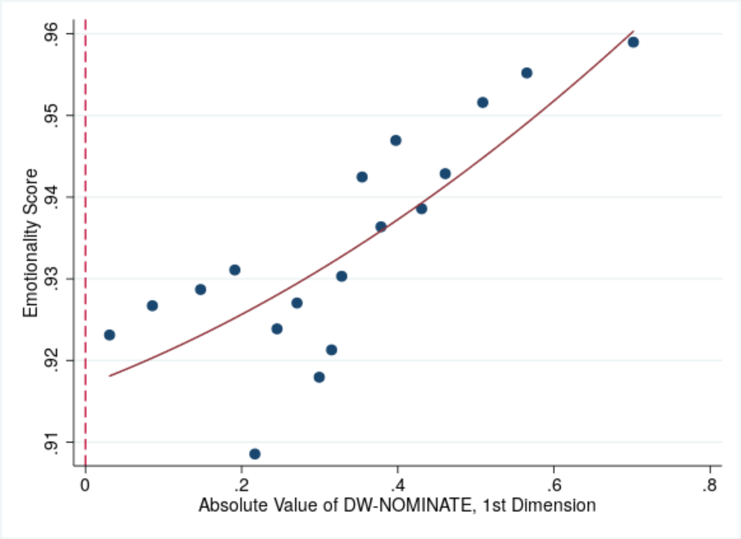


Relation to Congressman Characteristics

	(1)	(2)	(3)	(4)	(5)
		Estimated Effect on Emotionality Score			
Female	0.0516**		0.0475**	0.0489**	0.0301**
	(0.00651)		(0.00752)	(0.00645)	(0.00285)
Democrat		0.00638*	0.00502*	0.00315	0.00409**
		(0.00254)	(0.00252)	(0.00250)	(0.00136)
Female × Democrat			0.00405		
			(0.0110)		
Black				0.0282*	0.0208**
				(0.0117)	(0.00645)
Hispanic				0.0149	0.0133*
				(0.0113)	(0.00613)
Catholic				0.00953*	0.00567*
				(0.00442)	(0.00235)
Jewish				0.0109	0.00272
				(0.00780)	(0.00356)
Chamber-Year FE	X	X	X	X	X
Topic FE					X
N	5869780	5869780	5869780	5869780	5839095
adj. R^2	0.062	0.060	0.062	0.063	0.479

Std err. in parens, clustered by speaker. + p < .1, * p < .05, ** p<0.01.

Ideologically Extreme Politicians are More Emotive



Reviewing Gennaro and Ash (2021)

1. What is the methodological problem?
2. What is the substantive research question?

Reviewing Gennaro and Ash (2021)

1. What is the methodological problem?
2. What is the substantive research question?
3. Measurement:
   (a) What is the latent factor that ideally could be measured?
   (b) What is actually being measured?

Reviewing Gennaro and Ash (2021)

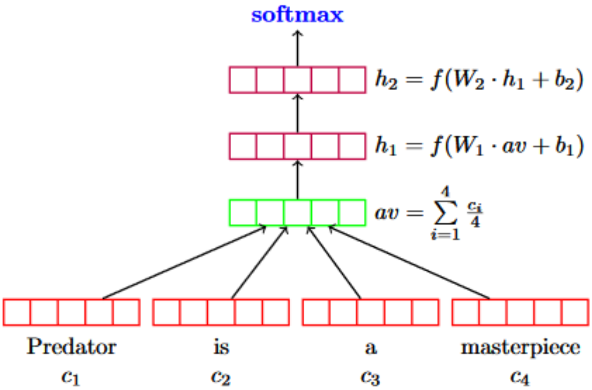
1. What is the methodological problem?
2. What is the substantive research question?
3. Measurement:
   (a) What is the latent factor that ideally could be measured?
   (b) What is actually being measured?
   (c) What are some gaps between (a) and (b)?
   (d) How do the validation steps address those gaps?

Deep Averaging Network (Iyyer et al 2015)

- Similar to the previous aggregated word embedding methods, but embeddings are learned during training:

Deep Averaging Network (Iyyer et al 2015)

- Similar to the previous aggregated word embedding methods, but embeddings are learned during training:



1. Trainable embedding layer for words, initialized with pre-trained embeddings
2. Average the embeddings, with dropout (sometimes words left out of average)
3. Average embedding fed into MLP with multiple hidden layers
4. MLP outputs used for classification or regression

fastText: Hashed N-Gram Embeddings (Joulin et al 2016)

Combines the Iyyer et al (2015) approach with the hashing n-gram vectorizer.

fastText: Hashed N-Gram Embeddings (Joulin et al 2016)

Combines the Iyyer et al (2015) approach with the hashing n-gram vectorizer.

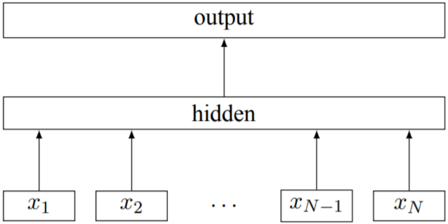


Figure 1: Model architecture of `fastText` for a sentence with N ngram features $x_1, \ldots, x_N$. The features are embedded and averaged to form the hidden variable.

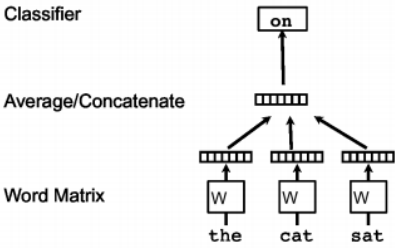
1. Allocate $n_w \approx 10$ million rows to embedding matrix.
2. Assign n-grams to embedding indexes with hashing function.
3. sentence embedding = average of n-gram embeddings
4. send to dense hidden layer(s)
5. send to output (e.g. classifier / regressor).

- Captures the local predictive power of n-grams without building vocabulary or costly training of CNN.

Outline

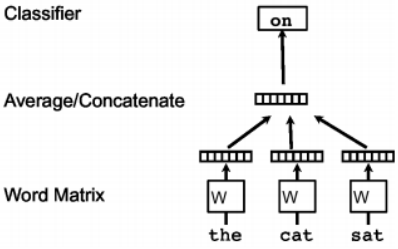
Doc2Vec (Le and Mikolov)

- Recall that Word2Vec trains word embeddings to predict a word given neighboring context words:

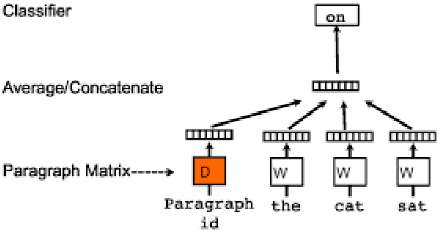


Doc2Vec (Le and Mikolov)

- Recall that Word2Vec trains word embeddings to predict a word given neighboring context words:



- Doc2Vec augments Word2Vec with a categorical embedding for the document (e.g. paragraph):



Doc2Vec on Wikipedia (Dai, Olah, and Le 2015)

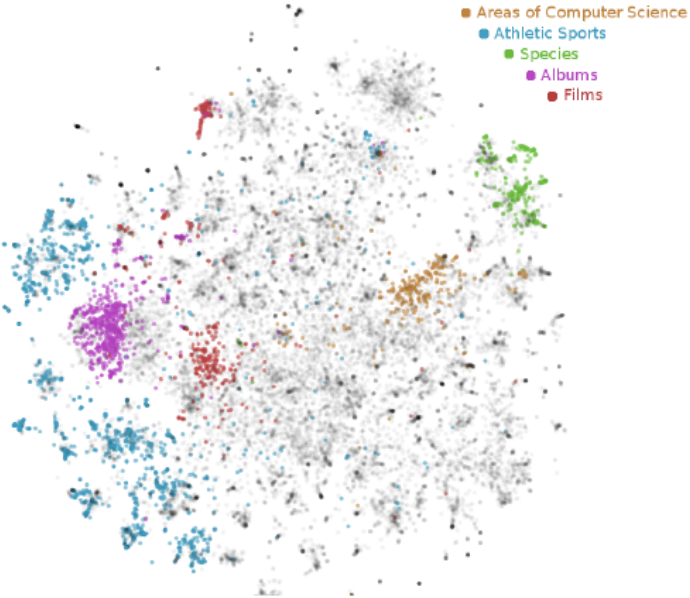
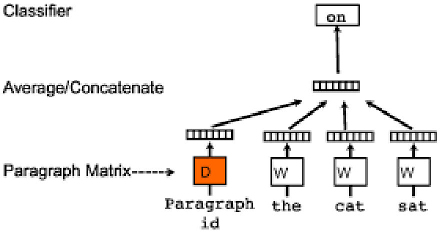


Figure 3: Visualization of Wikipedia paragraph vectors using t-SNE.

Vectorizing New Documents



- A new document that wasn't in training does not have a vector.
- Document inference step:
  - freeze word embeddings in input layer and in output layer.
  - learn embedding for new document to predict sampled words in new document.

Document Embeddings Geometry

- With topic models, each dimension has a topical interpretation.
- With document embeddings, a direction (might) have a topical interpretation.

Document Embeddings Geometry

- With topic models, each dimension has a topical interpretation.
- With document embeddings, a direction (might) have a topical interpretation.
- Analogous with word embeddings, directions in document embedding capture analogous dimensions of documents:

Table 2: Wikipedia nearest neighbours

(a) Wikipedia nearest neighbours to "Lady Gaga" using Paragraph Vectors. All articles are relevant.

Article	Cosine Similarity
Christina Aguilera	0.674
Beyonce	0.645
Madonna (entertainer)	0.643
Artpop	0.640
Britney Spears	0.640
Cyndi Lauper	0.632
Rihanna	0.631
Pink (singer)	0.628
Born This Way	0.627
The Monster Ball Tour	0.620

(b) Wikipedia nearest neighbours to "Lady Gaga" - "American" + "Japanese" using Paragraph Vectors. Note that Ayumi Hamasaki is one of the most famous singers, and one of the best selling artists in Japan. She also has an album called "Poker Face" in 1998.

Article	Cosine Similarity
Ayumi Hamasaki	0.539
Shoko Nakagawa	0.531
Izumi Sakai	0.512
Urbangarde	0.505
Ringo Sheena	0.503
Toshiaki Kasuga	0.492
Chihiro Onitsuka	0.487
Namie Amuro	0.485
Yakuza (video game)	0.485
Nozomi Sasaki (model)	0.485

Doc2Vec for Judicial Opinions (Ash and Chen 2018)

- Corpus: 300,000 cases from U.S. Circuit Courts, 1870-2010.
- Produce document vectors for each case to understand differences between judges and courts.

Doc2Vec for Judicial Opinions (Ash and Chen 2018)

- Corpus: 300,000 cases from U.S. Circuit Courts, 1870-2010.
- Produce document vectors for each case to understand differences between judges and courts.
- De-mean vectors by group (court, topic, or year) to extract relevant information:
  - de-mean by topic-year to distinguish courts.
  - de-mean by court-topic to distingush years.
  - de-mean by court-year to distinguish topics.

Figure 1: Centered by Topic-Year, Averaged by Judge, Labeled by Court

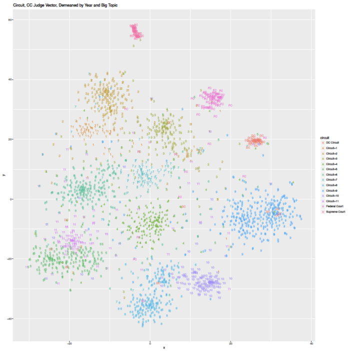


Figure 2: Centered by Court-Topic, Averaged by Court-Year, Labeled by Decade

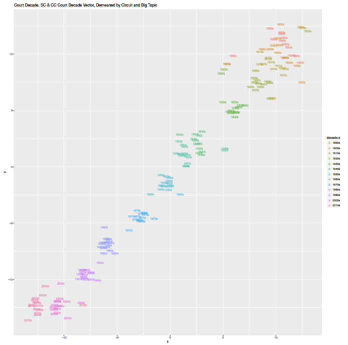
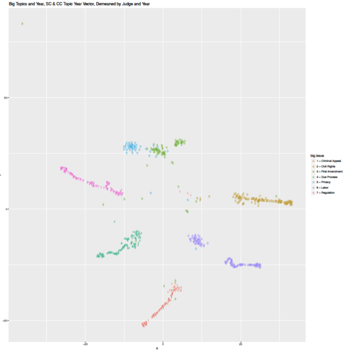


Figure 3: Centered by Judge-Year, Averaged by Topic-Year, Labeled by Topic



Outline

Sentence embeddings

- Document embedding allows us to think of documents as collections of sentences or paragraphs, rather than of words or phrases.
  - would be too high-dimensional to represent each sentence as a vector of word/phrase frequencies.

Sentence embeddings

- Document embedding allows us to think of documents as collections of sentences or paragraphs, rather than of words or phrases.
  - would be too high-dimensional to represent each sentence as a vector of word/phrase frequencies.
- e.g., sentence clustering:
  - run k-means clustering on the dataset of sentences embeddings, then represent documents as counts/frequencies over the sentence clusters.
  - show example sentences to interpret the clusters

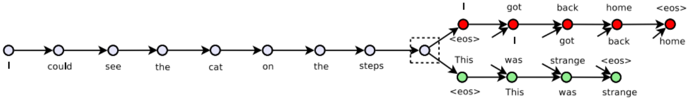
Sentence embeddings

- Document embedding allows us to think of documents as collections of sentences or paragraphs, rather than of words or phrases.
  - would be too high-dimensional to represent each sentence as a vector of word/phrase frequencies.
- e.g., sentence clustering:
  - run k-means clustering on the dataset of sentences embeddings, then represent documents as counts/frequencies over the sentence clusters.
  - show example sentences to interpret the clusters
- Sentence mover distance (Clark et al 2019):
  - adapt the idea of word mover distance to sentences.
  - find minimal cost of moving a set of sentence embeddings in document A to co-locate wth a set of sentence embeddings in document B.

Skip-Thought Embeddings (Kiros et al 2015)

Skip-Thought Embeddings (Kiros et al 2015)

- Same intuition as skip-gram embeddings in word2vec → produce sentence embeddings for a sentence prediction task.
  - gated recurrent encoder vectorizes a sentence, and the decoder tries to reproduce the next sentence.



  - uses negative sampling: produce embeddings to guess whether two sentences are in the same paragraph.

Query and nearest sentence
he ran his hand inside his coat , double-checking that the unopened letter was still there . he slipped his hand between his coat and his shirt , where the folded copies lay in a brown envelope .
im sure youll have a glamorous evening , she said , giving an exaggerated wink . im really glad you came to the party tonight , he said , turning to her .
although she could tell he had n't been too invested in any of their other chitchat , he seemed genuinely curious about this . although he had n't been following her career with a microscope , he 'd definitely taken notice of her appearances .
an annoying buzz started to ring in my ears , becoming louder and louder as my vision began to swim . a weighty pressure landed on my lungs and my vision blurred at the edges , threatening my consciousness altogether .
if he had a weapon , he could maybe take out their last imp , and then beat up errol and vanessa . if he could ram them from behind , send them sailing over the far side of the levee , he had a chance of stopping them .
then , with a stroke of luck , they saw the pair head together towards the portaloos . then , from out back of the house , they heard a horse scream probably in answer to a pair of sharp spurs digging deep into its flanks .
" i 'll take care of it , " goodman said , taking the phonebook . " i 'll do that , " julia said , coming in .
he finished rolling up scrolls and , placing them to one side , began the more urgent task of finding ale and tankards . he righted the table , set the candle on a piece of broken plate , and reached for his flint , steel , and tinder .

Table 2: In each example, the first sentence is a query while the second sentence is its nearest neighbour. Nearest neighbours were scored by cosine similarity from a random sample of 500,000 sentences from our corpus.

Universal Sentence Encoder

```python
import tensorflow_hub as hub

embed = hub.Module("https://tfhub.dev/google/"
    "universal-sentence-encoder/1")

embedding = embed([
    "The quick brown fox jumps over the lazy dog."])
```

Listing 1: Python example code for using the universal sentence encoder.

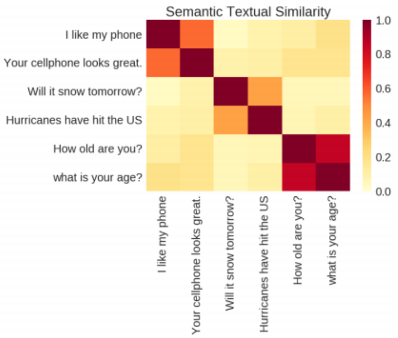


Figure 1: Sentence similarity scores using embeddings from the universal sentence encoder.

- Architecture:
    - Deep Averaging Network with embedded words and bigrams.

Universal Sentence Encoder

```
import tensorflow_hub as hub

embed = hub.Module("https://tfhub.dev/google/"
    "universal-sentence-encoder/1")

embedding = embed([
    "The quick brown fox jumps over the lazy dog."])
```

Listing 1: Python example code for using the universal sentence encoder.

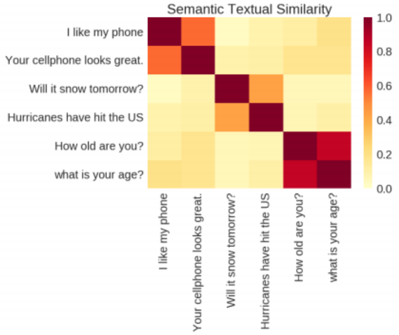


Figure 1: Sentence similarity scores using embeddings from the universal sentence encoder.

- Architecture:
    - Deep Averaging Network with embedded words and bigrams.
- Multiple pre-training objectives:
    - Identifying co-occuring sentences (as in skip thought vectors)
    - Identifying message-response pairs (Henderson et al 2017)
    - Some supervised learning tasks (see Cer et al 2018).

InferSent

- Train a bidirectional LSTM on Stanford Natural Language Inference task:
  - classifying 570K sentence pairs by entailment, contradiction, and neutral.
- The resulting sentence embeddings do better than skip-thought vectors on transfer learning tasks.

Multingual Encoders

- The multilingual sentence encoder (**MUSE**) expands the USE model to sixteen languages, in a single embedding model.
  - Trained on a similar array of tasks in all languages, so that it can be used out-of-the-box.

Multingual Encoders

- The multilingual sentence encoder (**MUSE**) expands the USE model to sixteen languages, in a single embedding model.
  - Trained on a similar array of tasks in all languages, so that it can be used out-of-the-box.
- Facebook's LASER encoder produces vectors for 90 languages with a single model.
  - bidirectional LSTM architecture
  - trained on multilingual machine translation task

Outline

Document Embeddings

- Aggregated word embeddings
- Doc2Vec
- Sentence Embeddings
- StarSpace

Attention

- Transformers: Overview
- Self-Attention

StarSpace: Embed Anything (Wu et al 2018)

Generalize two key embedding ingredients from NLP to much broader set of tasks:

StarSpace: Embed Anything (Wu et al 2018)

Generalize two key embedding ingredients from NLP to much broader set of tasks:

1. aggregate embeddings across words or phrases by document $\rightarrow$ aggregate embeddings across features by entity
2. negative sampling of co-locating words vs random words $\rightarrow$ negative sampling of related entities vs unrelated entities

Entities and Features

- **features** are categorical variables.
  - learn $n_F \times n_E$ embedding matrix F with n_F features and embedding dimension n_E.
- **entities** are bags of features:
  - for entity consisting of features $a = \{1, 2, ..., i, ...\}$, sum over feature embeddings:

$$\vec{a} = \sum_{i \in a} F_i$$

    where F_i indicates the associated row of F.

Entities and Features

- **features** are categorical variables.
  - learn $n_F \times n_E$ embedding matrix F with n_F features and embedding dimension n_E.
- **entities** are bags of features:
  - for entity consisting of features $a = \{1, 2, ..., i, ...\}$, sum over feature embeddings:

$$\vec{a} = \sum_{i \in a} F_i$$

    where F_i indicates the associated row of F.
- Then by construction, entities and features are in the same space.

StarSpace Negative Sampling Objective

- For entity a selected at current training batch:
  - positive sample: related entity b (e.g. two sentences from the same document).
  - negative samples: k unrelated entities $b_1^-, \ldots b_k^-$ (e.g. sentences in other documents).

StarSpace Negative Sampling Objective

- For entity a selected at current training batch:
  - positive sample: related entity b (e.g. two sentences from the same document).
  - negative samples: k unrelated entities $b_1^-, \ldots b_k^-$ (e.g. sentences in other documents).
- Compute vectors $\vec{a} = \sum_{i \in a} F_i$, $\vec{b}$, $\vec{b}_1^-, \ldots, \vec{b}_k^-$
- Compute cosine similiarities $\text{sim}(\vec{a}, \vec{b})$,$\text{sim}(\vec{a}, \vec{b}_1^-)$, ..., $\text{sim}(\vec{a}, \vec{b}_k^-)$,

StarSpace Negative Sampling Objective

- For entity a selected at current training batch:
  - positive sample: related entity b (e.g. two sentences from the same document).
  - negative samples: k unrelated entities $b_1^-, \ldots b_k^-$ (e.g. sentences in other documents).
- Compute vectors $\vec{a} = \sum_{i \in a} F_i$, $\vec{b}$, $\vec{b}_1^-, \ldots, \vec{b}_k^-$
- Compute cosine similiarities $\mathrm{sim}(\vec{a}, \vec{b})$,$\mathrm{sim}(\vec{a}, \vec{b}_1^-)$, ..., $\mathrm{sim}(\vec{a}, \vec{b}_k^-)$,
- Ranking loss $\max\{0, \mu - \mathrm{sim}(\vec{a},\vec{b})\}$
  - reward for $\mathrm{sim}(\vec{a}, \vec{b})$ getting higher rank relative to the negative samples, penalty for lower rank.

Learning (unsupervised) Sentence Embeddings

Directly/Optimally learn sentence embed

Select a pair of sents (s1, s2) from the same doc:

a: s1
b: s2
b-: sampled from sents coming from other docs

- but StarSpace can be used for anything.
- the trained model can provide similarities between entities, between features, and between entities and features.

No social science papers with StarSpace

But many opportunities:

- embed judicial opinions as bundles of citations
- embed academic articles as bundles of citations
- embed politicians as bundles of roll call votes

Outline

Outline

Attention is All you Need: Transformers

- Since a 2017 paper (Vaswani et al 2017), deep learning for NLP has been transformed by a new class of models: **transformers.**
  - replaces recurrence or convolutions with ***attention***.

Attention is All you Need: Transformers

- Since a 2017 paper (Vaswani et al 2017), deep learning for NLP has been transformed by a new class of models: **transformers.**
  - replaces recurrence or convolutions with ***attention***.
- "Attention" is a type of soft convolutional filter, which provides a weighted aggregation over a sequence, with task-relevant words up-weighted by the attention filter.

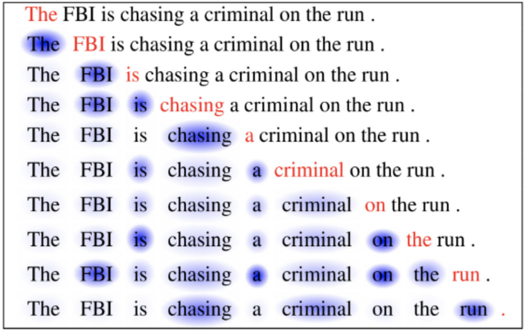


Fig. 6. The current word is in red and the size of the blue shade indicates the activation level. (Image source: Cheng et al., 2016)

- A transformer consists of multiple "transformer layers", which consist of multiple parallel attention filters.

OPENAI'S NEW MULTITALENTED AI WRITES, TRANSLATES, AND SLANDERS

A step forward in AI text-generation that also spells trouble

By James Vincent | Feb 14, 2019, 12:00pm EST

Howard, co-founder of Fast.AI agrees. "I've been trying to warn people about this for a while," he says. "We have the technology to totally fill Twitter, email, and the web up with reasonable-sounding, context-appropriate prose, which would drown out all other speech and be impossible to filter."

https://transformer.huggingface.co/doc/distil-gpt2

GPT and BERT are Pre-Trained Transformer Models

- **GPT = "Generative Pre-Trained Transformer"**:
  - train transformer to predict the next word at the end of a sequence.
  - now in it's third version (GPT-3)

GPT and BERT are Pre-Trained Transformer Models

- **GPT = "Generative Pre-Trained Transformer"**:
  - train transformer to predict the next word at the end of a sequence.
  - now in it's third version (GPT-3)
  - notably good for text generation (language modeling)

GPT and BERT are Pre-Trained Transformer Models

- **GPT = "Generative Pre-Trained Transformer"**:
  - train transformer to predict the next word at the end of a sequence.
  - now in it's third version (GPT-3)
  - notably good for text generation (language modeling)
- **BERT = "Bidrectional Encoder Representations from Transormers"**:
  - train transformer to predict left-out words in the middle of a sequence.

GPT and BERT are Pre-Trained Transformer Models

- **GPT = "Generative Pre-Trained Transformer"**:
  - train transformer to predict the next word at the end of a sequence.
  - now in it's third version (GPT-3)
  - notably good for text generation (language modeling)
- **BERT = "Bidrectional Encoder Representations from Transormers"**:
  - train transformer to predict left-out words in the middle of a sequence.
  - the resulting document embeddings contain most (perhaps all?) of the relevant information in short language snippets.
    - blew away all the NLP baselines (e.g. semantic role labeling, question-answering, entailment, etc.) when it came out in 2018.

GPT and BERT are Pre-Trained Transformer Models

- **GPT = "Generative Pre-Trained Transformer"**:
  - train transformer to predict the next word at the end of a sequence.
  - now in it's third version (GPT-3)
  - notably good for text generation (language modeling)
- **BERT = "Bidirectional Encoder Representations from Transormers"**:
  - train transformer to predict left-out words in the middle of a sequence.
  - the resulting document embeddings contain most (perhaps all?) of the relevant information in short language snippets.
    - blew away all the NLP baselines (e.g. semantic role labeling, question-answering, entailment, etc.) when it came out in 2018.
- immediately relevant use cases for our purpose:
  - many pre-trained models, e.g. for sentiment classification
  - BERT model can be fine-tuned to quickly get optimal results for many text classification tasks.

Shortcut: Using BERT-Based Pre-Trained Models

Shortcut: Using BERT-Based Pre-Trained Models

```
from transformers import pipeline
sentiment_analysis = pipeline("sentiment-analysis")

pos_text = "I enjoy studying computational algorithms."
neg_text = "I dislike sleeping late everyday."

pos_sent = sentiment_analysis(pos_text)[0]
print(pos_sent['label'],0 pos_sent['score'])

neg_sent = sentiment_analysis(neg_text)[0]
print(neg_sent['label'], neg_sent['score'])
```

- also straightforward to fine-tune BERT for your own classification tasks.
- see week 9's notebooks for full details / explanation.

Outline

Self-Attention – the fundamental computation underlying transformers

- Consider a sequence of tokens with fixed length n_L, $\{w_1, ..., w_i, ..., w_{n_L}\}$
- We have word embedding vectors $x_i = E(w_i)$ with dimension n_E, producing a sequence of vectors

$$\{x_1, ..., x_i, ..., x_{n_L}\}$$

- In previous models, the sequence $x_{1:n_L}$ could be flattened to an $n_L n_E$-dimensional vector and piped to the hidden layers for use in the task, e.g. sentiment classification.

Self-Attention – the fundamental computation underlying transformers

- Consider a sequence of tokens with fixed length n_L, $\{w_1, ..., w_i, ..., w_{n_L}\}$
- We have word embedding vectors $x_i = E(w_i)$ with dimension n_E, producing a sequence of vectors

$$\{x_1, ..., x_i, ..., x_{n_L}\}$$

- In previous models, the sequence $x_{1:n_L}$ could be flattened to an $n_L n_E$-dimensional vector and piped to the hidden layers for use in the task, e.g. sentiment classification.
- A **self-attention layer** transforms $x_{1:n_L}$ into a second sequence $h_{1:n_L}$, where

$$h_i = \sum_{j=1}^{n_L} a(x_i, x_j) x_j$$

  - where $a(\cdot)$ is an attention function such that $a(\cdot) \geq 0$, $\sum a(\cdot) = 1$.
  - $\rightarrow$ each h_i becomes a weighted average of the whole sequence.
- $h_{1:n_L}$ is flattened and piped to the network's hidden layers, rather than $x_{1:n_L}$.

Basic Self-Attention

Setup:

1. Sequence of tokens $\{w_1, ..., w_i, ..., w_{n_L}\}$
2. Sequence of (trainable) embedding vectors $\{x_1, ..., x_i, ..., x_{n_L}\}$
3. Sequence of attention-transformed vectors $\{h_1, ..., h_i, ..., h_{n_L}\}$ with

$$h_i = \sum_{j=1}^{n_L} a(x_i, x_j) x_j$$

Basic Self-Attention

Setup:

1. Sequence of tokens $\{w_1, ..., w_i, ..., w_{n_L}\}$
2. Sequence of (trainable) embedding vectors $\{x_1, ..., x_i, ..., x_{n_L}\}$
3. Sequence of attention-transformed vectors $\{h_1, ..., h_i, ..., h_{n_L}\}$ with

$$h_i = \sum_{j=1}^{n_L} a(x_i, x_j) x_j$$

Basic self-attention specifies

$$a(x_i, x_j) = \frac{\exp(x_i \cdot x_j)}{\sum_{k=1}^{n_L} \exp(x_i \cdot x_k)}$$

- the dot-product $x_i \cdot x_j$, normalized with softmax such that $\sum_j a(\cdot) = 1$.

Basic Self-Attention

Setup:

1. Sequence of tokens $\{w_1, ..., w_i, ..., w_{n_L}\}$
2. Sequence of (trainable) embedding vectors $\{x_1, ..., x_i, ..., x_{n_L}\}$
3. Sequence of attention-transformed vectors $\{h_1, ..., h_i, ..., h_{n_L}\}$ with

$$h_i = \sum_{j=1}^{n_L} a(x_i, x_j) x_j$$

Basic self-attention specifies

$$a(x_i, x_j) = \frac{\exp(x_i \cdot x_j)}{\sum_{k=1}^{n_L} \exp(x_i \cdot x_k)}$$

- the dot-product $x_i \cdot x_j$, normalized with softmax such that $\sum_j a(\cdot) = 1$.
- Putting it together:

$$h_i = \sum_{j=1}^{n_L} \frac{\exp(x_i \cdot x_j)}{\sum_{k=1}^{n_L} \exp(x_i \cdot x_k)} x_j$$

- The basic self-attention transformation

$$h_i = \sum_{j=1}^{n_L} \frac{\exp(x_i \cdot x_j)}{\sum_{k=1}^{n_L} \exp(x_i \cdot x_k)} x_j$$

is the foundational ingredient of transformers.

- The basic self-attention transformation

$$h_i = \sum_{j=1}^{n_L} \frac{\exp(x_i \cdot x_j)}{\sum_{k=1}^{n_L} \exp(x_i \cdot x_k)} x_j$$

  is the foundational ingredient of transformers.

Note the following simplifications:

- **basic self-attention has no learnable parameters**.
  - self-attention works indirectly through the word embeddings (more next slide)
- **basic self-attention ignores word order.**

- The basic self-attention transformation

$$h_i = \sum_{j=1}^{n_L} \frac{\exp(x_i \cdot x_j)}{\sum_{k=1}^{n_L} \exp(x_i \cdot x_k)} x_j$$

  is the foundational ingredient of transformers.

Note the following simplifications:

- **basic self-attention has no learnable parameters**.
  - self-attention works indirectly through the word embeddings (more next slide)
- **basic self-attention ignores word order.**

The big initial gain from transformers, relative to RNNs, came from basic self-attention.

- The successful models (e.g. BERT, GPT) do add parameters and word order information to $a(\cdot)$ (to be discussed in Week 9 lecture).

Why self-attention works

- Consider a sentence

$$\text{the, cat, walks, on, the, street}$$

  with embeddings

$$\boldsymbol{x}_{\text{the}}, \boldsymbol{x}_{\text{cat}}, \boldsymbol{x}_{\text{walks}}, \boldsymbol{x}_{\text{on}}, \boldsymbol{x}_{\text{the}}, \boldsymbol{x}_{\text{street}}$$

- Feeding this sentence into the self-attention layer produces

$$\boldsymbol{h}_{\text{the}}, \boldsymbol{h}_{\text{cat}}, \boldsymbol{h}_{\text{walks}}, \boldsymbol{h}_{\text{on}}, \boldsymbol{h}_{\text{the}}, \boldsymbol{h}_{\text{street}}$$

  where $\boldsymbol{h}_i = \sum_{j=1}^{n} \frac{\exp(\boldsymbol{x}_i \cdot \boldsymbol{x}_j)}{\sum_k \exp(\boldsymbol{x}_i \cdot \boldsymbol{x}_k)} \cdot \boldsymbol{x}_j$.

Why self-attention works

- Consider a sentence

  the, cat, walks, on, the, street

  with embeddings

  $$\boldsymbol{x}_{\text{the}}, \boldsymbol{x}_{\text{cat}}, \boldsymbol{x}_{\text{walks}}, \boldsymbol{x}_{\text{on}}, \boldsymbol{x}_{\text{the}}, \boldsymbol{x}_{\text{street}}$$

- Feeding this sentence into the self-attention layer produces

  $$\boldsymbol{h}_{\text{the}}, \boldsymbol{h}_{\text{cat}}, \boldsymbol{h}_{\text{walks}}, \boldsymbol{h}_{\text{on}}, \boldsymbol{h}_{\text{the}}, \boldsymbol{h}_{\text{street}}$$

  where $\boldsymbol{h}_i = \sum_{j=1}^{n} \frac{\exp(\boldsymbol{x}_i \cdot \boldsymbol{x}_j)}{\sum_k \exp(\boldsymbol{x}_i \cdot \boldsymbol{x}_k)} \cdot \boldsymbol{x}_j$.
- Embedding layer will learn vectors $\boldsymbol{x}$ that tend to have **attention dot products** that contribute to the task at hand.
  - For example, in predicting the next sentence, stopwords like "the" will not be helpful.
  - the learned embedding $\boldsymbol{x}_{\text{the}}$ will tend to have a low or negative dot product with more informative words.
- related to GloVe objective, which rewards higher dot product for words that appear in similar contexts.