Text Analytics & Business Application

Text Representation - Embeddings

Qinglai He
Department of Operations and Information Management
Wisconsin School of Business

WISCONSIN SCHOOL OF BUSINESS

IC4 Discussions

- Why naïve bayes did not give us good results?
- Why using higher grams (bigram or trigram only) leads to worse performance?
- Using more data helps with the performance.

Text Representation (Cont.)

- ~~Vector Space Model~~
- ~~Basic Vectorization Approaches~~
  - ~~One-Hot Encoding~~
  - ~~Bag of Words~~
  - ~~Bag of N-Grams~~
  - ~~TF-IDF~~
- Distributed Representation
  - Word embedding
  - Document embedding
  - BERT

Distributed Representations

- If we wanted to represent a new shape with a basic vectorization approaches, we would have to **increase the dimensionality**.
- To overcome this limitation, methods to learn low-dimensional representations were devised.
- Used neural network architectures to create **dense**, **low-dimensional** representations of words and texts.

Some Prior Knowledge

- Distributional **similarity**
  - An idea that the meaning of a word can be understood from the context in which the word appears.
  - For example: “NLP rocks.” The literal meaning of the word “rocks” is “stones,” but from the context, it’s used to refer to something good and fashionable.

- Distributional **hypothesis**
  - If two words often occur in similar context, then their corresponding representation vectors must also be close to each other.
  - For example, the English words “dog” and “cat” occur in similar contexts. Thus, according to the distributional hypothesis, there must be a strong similarity between the meanings of these two words.

1. Word Embeddings

- What does it mean when we say a text representation should capture "distributional similarities between words"?
- For example:
  - If we're given the word "USA," distributionally similar words could be other countries (e.g., Canada, Germany, India, etc.) or cities in the USA.
  - If we're given the word "beautiful," words that share some relationship with this word (e.g., synonyms, antonyms) could be considered distributionally similar words.

Another Example

- There is some *tezgüino*
- A jar of *tezgüino* is on the table
- Everybody likes *tezgüino*
- I’ll have a drink of *tezgüino*
- We make *tezgüino* out of corn, but we do not distill it.

What is *tezgüino*?

Word Embedding Intuition

- Goal: Looking for a latent space, which preserves the semantics as much as possible.
- An assumption is that words which often have the same contexts tend to be similar in semantics
  - Word co-occurrence
  - Examples: dog vs. cat, coffee vs. tea
- Text representation: a word ***w*** will be represented as a vector
- How to define "context"
  - The document it appeared
  - Nearby words
  - …

Pre-trained Word Embeddings

- Training your own word embeddings is a pretty expensive process, however, using pre-trained word embeddings often suffices
- What are pre-trained word embeddings?
  - Someone has done the hard work of training word embeddings on a large corpus, such as Wikipedia, news articles
- Popular pre-train word embedding & extensions:
  - **Word2Vec**: a common method to generate word embeddings.
  - **GloVe**
  - **fastText**

(1) Word2vec

- In 2013, a seminal work by Mikolov et al. showed that their neural network–based word representation model known as "Word2vec," based on "distributional similarity," can capture word analogy relationships such as:

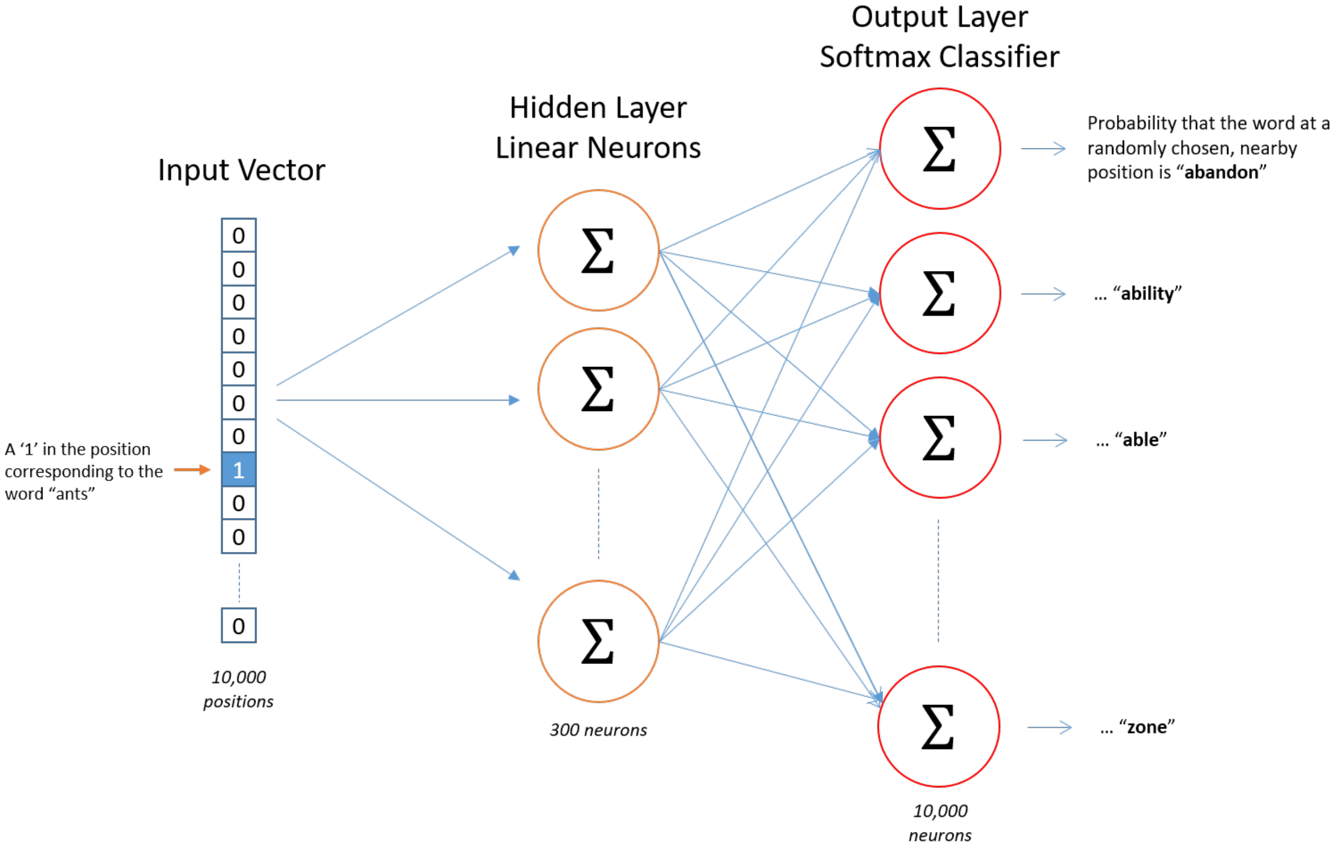
$$\text{King} - \text{Man} + \text{Woman} \approx \text{Queen}$$

- Word2vec ensures that the learned word representations are low dimensional
- Such representations make ML tasks more tractable and efficient
- Word2vec led to a lot of work (both pure and applied) in the direction of learning text representations using neural networks

Efficient Estimation of Word Representations in Vector Space (original word2vec paper)

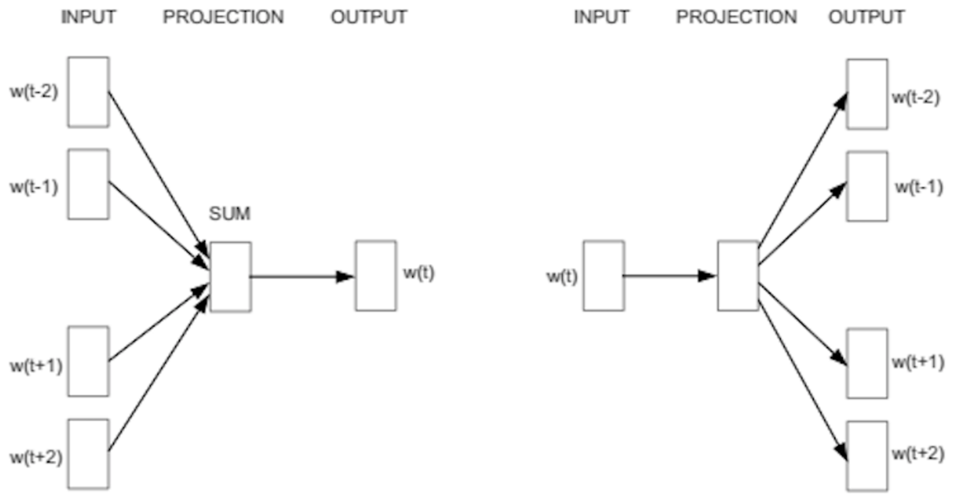
Neural network architecture

- Input layer:
  - one-hot input vector
- Hidden layer:
  - Dense layer
  - word embeddings (weights) that we are going to learn
- Output layer:
  - Softmax probability



Two Main Architectures for Word2Vec

- Continuous bag of words (CBOW): given context words, can we predict the missing words?
  - For example: With sentence ”Have a good day”, we use [‘Have’, ‘good’, ‘day’] (context) to predict [‘a’] (word)
- Skip-Gram: given the word, can we predict its contexts?
  - For example: it takes the current word as an input and tries to accurately predict the words before and after this current word.

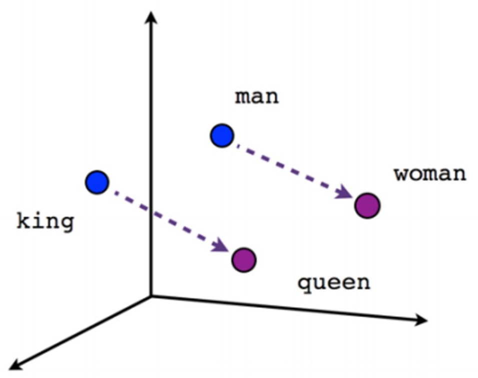


CBOW Skip-gram

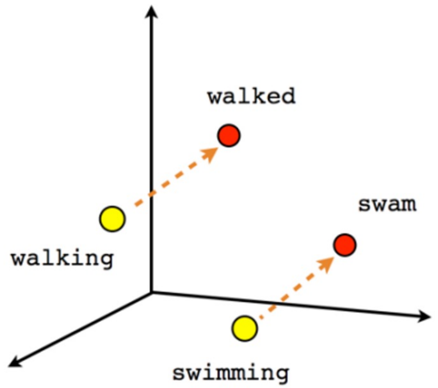
https://israelg99.github.io/2017-03-23-Word2Vec-Explained/

Interesting Properties of the Word Vectors

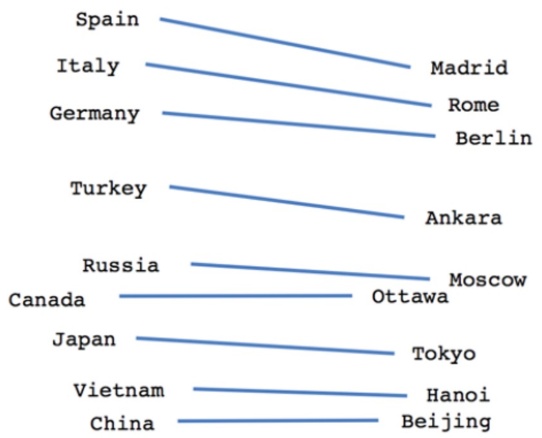
- Word embedding is able to capture multiple different degrees of similarity between words, such that semantic and syntactic patterns can be reproduced using vectors arithmetic.
- Famous example
  - Vector (‘king) – Vector (‘man) + Vector (‘woman) is close to Vector (‘queen’)



Male-Female



Verb tense



Country-Capital

https://israelg99.github.io/2017-03-23-Word2Vec-Explained/

Model evaluation

- Extrinsic and intrinsic evaluation

Extrinsic evaluation	Intrinsic evaluation
• Use word embeddings as input features to a downstream task and measure changes in performance metrics specific to that task • Con #1: Can take a long time to compute accuracy • Con #2: Unclear if the subsystem is the problem or its interaction or other subsystems	• Evaluation on syntactic or semantic relationships between words with score • Pro #1: Fast to compute • Con #1: Not clear if really helpful unless correlation to real task is established

Intrinsic Evaluation

Type of relationship	Word Pair 1		Word Pair 2	
Common capital city	Athens	Greece	Oslo	Norway
All capital cities	Astana	Kazakhstan	Harare	Zimbabwe
Currency	Angola	kwanza	Iran	rial
City-in-state	Chicago	Illinois	Stockton	California
Man-Woman	brother	sister	grandson	granddaughter
Adjective to adverb	apparent	apparently	rapid	rapidly
Opposite	possibly	impossibly	ethical	unethical
Comparative	great	greater	tough	tougher
Superlative	easy	easiest	lucky	luckiest
Present Participle	think	thinking	read	reading
Nationality adjective	Switzerland	Swiss	Cambodia	Cambodian
Past tense	walking	walked	swimming	swam
Plural nouns	mouse	mice	dollar	dollars
Plural verbs	work	works	speak	speaks

semantic

syntactic

- Answer the question:

$$\boxed{a : b \ :: \ c : ?}$$

man: women :: king: ?
good: better :: rough: ?

- Search with cosine similarity
  - $y = x_b - x_a + x_c$
  - $d = argmax_i \frac{y^T x_i}{||y||}$
  - Check if d is the answer in test data set

https://aclanthology.org/N13-1090.pdf https://arxiv.org/pdf/1301.3781.pdf%C3%AC%E2%80%94%20%C3%AC%E2%80%9E%C5%93

Results on the Word Analogy Task

Model	Vector Dimensionality	Training words	Accuracy [%]			Training time [days]
			Semantic	Syntactic	Total	
3 epoch CBOW	300	783M	15.5	53.1	36.1	1
3 epoch Skip-gram	300	783M	50.0	55.9	53.3	3
1 epoch CBOW	300	783M	13.8	49.9	33.6	0.3
1 epoch CBOW	300	1.6B	16.1	52.6	36.1	0.6
1 epoch CBOW	600	783M	15.4	53.3	36.2	0.7
1 epoch Skip-gram	300	783M	45.6	52.2	49.2	1
1 epoch Skip-gram	300	1.6B	52.2	55.1	53.8	2
1 epoch Skip-gram	600	783M	56.7	54.5	55.5	2.5

(2) Global Vectors (GloVe)

- Previous models review

	Global matrix factorization methods	Local context window methods
Models	• LSA, HAL (Lund & Burgess) • COALS, Hellinger-PCA (Rohde et al, Lebret & Collobert)	• Skip-gram/CBOW (Mikolov et al) • NNLM, HLBL, RNN (Bengio et al; Collobert & Weston; Huang et al; Mnih & Hinton)
Pros	• Fast training • Efficient usage of statistics	• Can capture complex patterns beyond word similarity • Generate improved performance on other tasks
Cons	• Capture complex patterns beyond word similarity • Generate improved performance on other tasks	• Scales with corpus size • Inefficient usage of statistics

- Intrinsic of GloVe: Combines the advantages of the two major model families
  - Utilize **global** co-occurrence matrix
  - Train only on the **nonzero** elements to leverage statistical information

https://nlp.stanford.edu/pubs/glove.pdf

Global Vectors (GloVe)

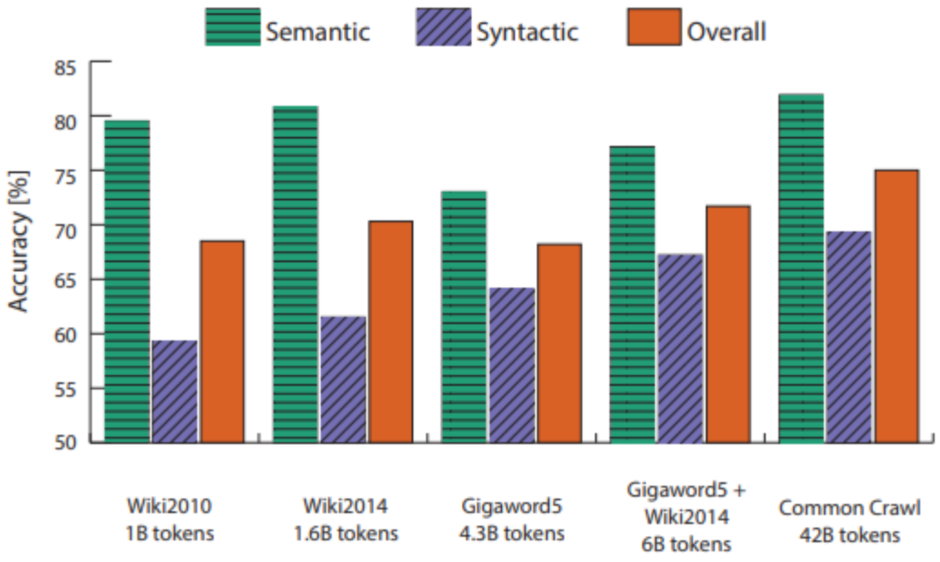
- **Results on the word analogy task**

Model	Dim.	Size	Sem.	Syn.	Tot.
SVD	300	6B	6.3	8.1	7.3
SVD-S	300	6B	36.7	46.6	42.1
SVD-L	300	6B	56.6	63.0	60.1
CBOW	300	6B	63.6	67.4	65.7
SG	300	6B	73.0	66.0	69.1
GloVe	300	6B	77.4	67.0	71.7

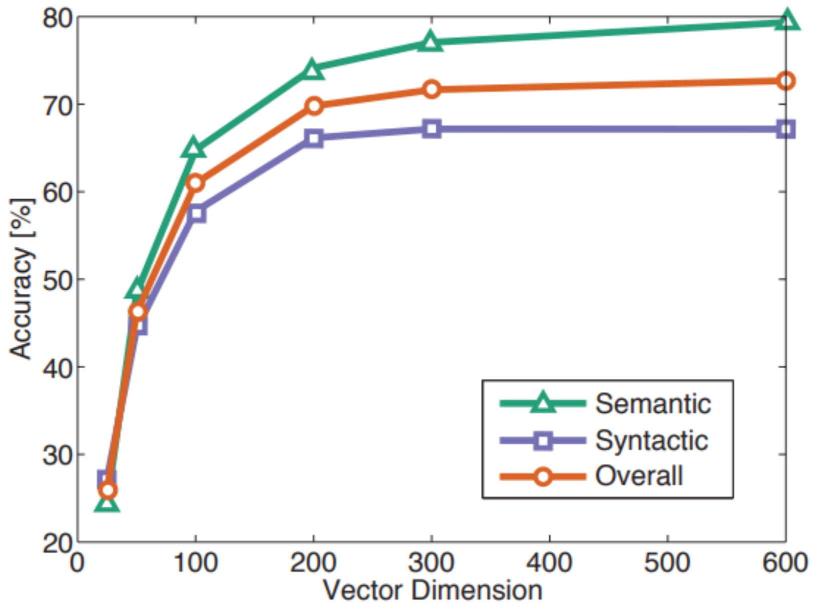
Global Vectors (GloVe)

- **More evaluations on GloVe**

- Evaluations regards training data
  - More data helps
  - Wikipedia is better than news text



- Evaluations regards dimension
  - Around 300 is a good choice



https://nlp.stanford.edu/pubs/glove.pdf

https://web.stanford.edu/class/archive/cs/cs224n/cs224n.1214/slides/cs224n-2021-lecture02-wordvecs2.pdf

(3) fastText

- Intrinsic of fastText: **Consider the internal structure of words**
  - An extension of the continuous skip-gram model with **subword** information
  - Use **character level information** to improve vector representations for morphologically rich languages
  - Learn representations for character n-grams, and represent words as the **sum of the n-gram vectors**
- Example
  - French or Spanish, most verbs have more than forty different inflected forms
  - Finnish language has fifteen cases for nouns

fastText

Many word forms that occur rarely (or not at all) in the training corpus
=
Difficult to learn good word representations for these words

https://arxiv.org/pdf/1607.04606.pdf https://fasttext.cc/

fastText

- Subword model: Represent word with sum of character n-gram vectors
  - Add “ < ” and “ > ” at the beginning and end of words
  - Also include the word itself
  - Associate a vector z_g representation to each n-gram g
  - Represent a word with the sum: $w = \sum_g z_g$
  - In practice, the set of n-grams N is restricted to the n-grams with 3 to 6 characters
- Example: 3-grams of *where*
  - **<where>**
  - *3-grams* plus *where* itself*: <wh, whe, her, ere, re>, <where>*

Recall Skip-Gram or CBOW model,
switch the word representation to w

https://arxiv.org/pdf/1607.04606.pdf

fastText

- **fastText vs. word2wec**
  - Word2Vec works on the word level, while fastText works on the character n-grams
  - Word2Vec cannot provide word vectors for out-of-vocabulary words, while fastText provides embeddings by summing up the n-grams vectors
  - fastText provides better embeddings for morphologically rich languages compared to word2vec
  - fastText uses the hierarchical classifier to train the model; hence it is faster than word2vec.

		sg	cbow	sisg (fastText-sg)
Czech (CS)	Semantic	25.7	**27.6**	27.5
	Syntactic	52.8	55.0	**77.8**
German (DE)	Semantic	66.5	**66.8**	62.3
	Syntactic	44.5	45.0	**56.4**
English (EN)	Semantic	**78.5**	78.2	77.8
	Syntactic	70.1	69.9	**74.9**
Italian (IT)	Semantic	52.3	**54.7**	52.3
	Syntactic	51.5	51.8	**62.7**

https://arxiv.org/pdf/1607.04606.pdf

https://www.analyticsvidhya.com/blog/2023/01/introduction-to-fasttext-embeddings-and-its-implication/#:~:text=Word2Vec%20cannot%20provide%20embeddings%20for,it%20is%20faster%20than%20word2vec.

2. Document Embeddings

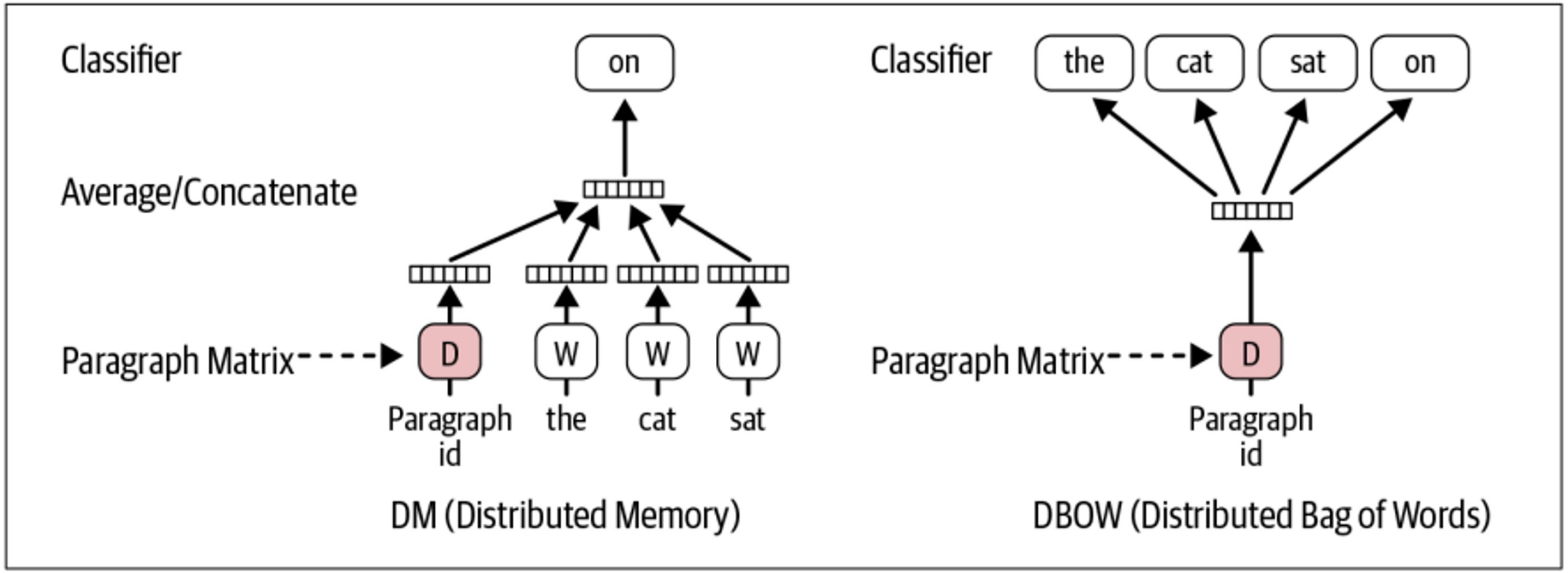
- Word2vec learned representations for words. To represent a document, we normally need to aggregate each word's embedding.
- Different from Word2Vec, **Doc2vec a**llows us to directly learn the representations for texts of arbitrary lengths (phrases, sentences, paragraphs, and documents) by taking the context of words in the text into account.

Doc2vec

- Doc2vec is based on the **paragraph vectors** framework and is implemented in genism.
- Similar to Word2vec in terms of its general architecture, but it also learns a "paragraph vector" that learns a representation for the full text (i.e., with words in context).
- When learning with a large corpus of many texts, the paragraph vectors are unique for a given text (where "text" can mean any piece of text of arbitrary length), while word vectors will be shared across all texts. The shallow neural networks used to learn Doc2vec embeddings are very similar to the CBOW and SkipGram architecture of Word2vec.

Doc2vec

- The two architectures are called:
  - Distributed memory (DM)
  - Distributed bag of words (DBOW)

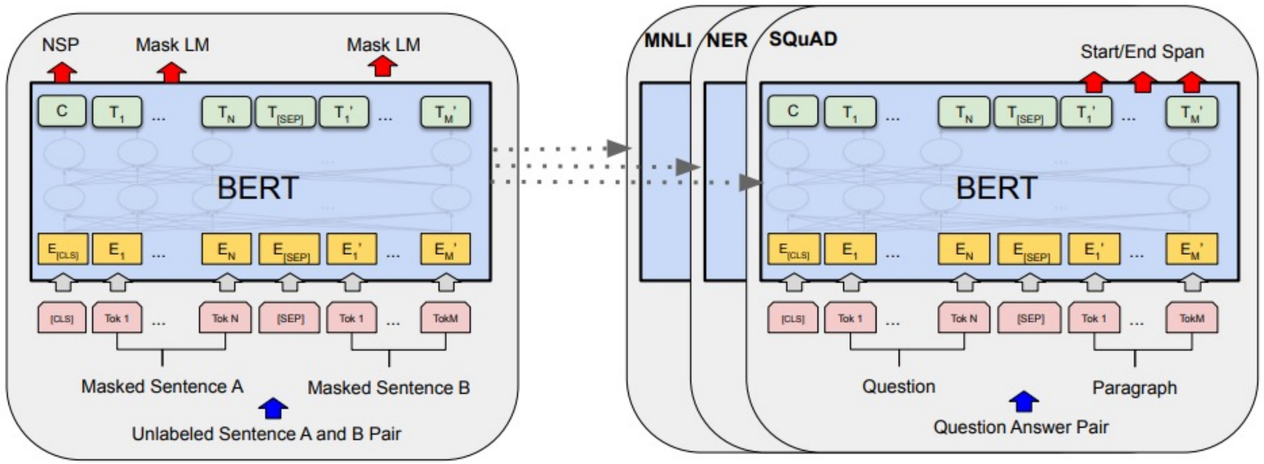


3. BERT

- **B**idirectional **E**ncoder **R**epresentations from **T**ransformers
- BERT came out in 2018
- It is designed to pretrain deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers
- It obtains new state-of-the-art results on 11 natural language processing tasks

Read more about BERT

- The original paper
- BERT Github
- The Illustrated BERT, ELMo, and co.
- BERT Explained: State of the art language model for NLP



Pre-training
Fine-Tuning

Pre-trained Models

- What is pre-trained models?
  - A pre-trained model is a model that’s trained on large datasets to accomplish a specific task, and it can be used as is or further fine-tuned to fit an application’s specific needs.
- Why use pre-trained models?
  - Reduced training time
  - Improved performance
  - Reduced risk of overfitting
- Two strategies for applying pre-trained language representations:

Feature-based	Fine-tuning
Use task-specific architectures that include the pre-trained representations as additional features	Introduce minimal task-specific parameters, and is trained on the downstream tasks by simply fine-tuning all pretrained parameters
ELMo	Generative Pre-trained Transformer (OpenAI GPT), **BERT**

Practical Advice

- Often in NLP, feeding a **good text representation** to an ordinary algorithm will get you much farther compared to applying a top-notch algorithm to an ordinary text representation.
- These days, one-hot encoding is seldom used. TF-IDF continues to be a popular representation scheme for many NLP tasks, especially the initial versions of the solution.
- If you're new to **embeddings**, always start by using **pre-trained word embeddings** in your project.
- If the overlap between corpus vocabulary and embedding vocabulary is less than 80%, we're unlikely to see good performance from our NLP model.


QUIZ

Q1. Which of the following is not word embedding?

A. Word2Vec
B. Doc2Vec
C. fastText
D. GloVe

Answer: B

QUIZ

Q2. Which of the following architectures is used in word2vec for generating word embeddings?

A. CBOW
B. Skip-gram
C. CNN
D. RNN

Answer: A, B

QUIZ

Q3. Which of the following embeddings might help mitigate the out-of-vocabulary issue?

A. Word2Vec
B. fastText
C. Doc2Vec
D. GloVe

Answer: B

Q4. Which technique is used to create embeddings for entire documents instead of individual words?

A. Rule-based approach
B. Word2Vec
C. Machine learning model
D. Doc2Vec

Answer: D

Q5. What is the primary objective of the skip-gram architecture in Word2Vec?

A. Predicting the context words given a target word
B. Predicting the target word given a set of context words
C. Generating word embeddings based on co-occurrence statistics
D. Classifying text documents into categories

Answer: A

Q6. Is the following statement true or false?
"Training our own embedding is preferred when many words in our documents cannot be found in the pre-trained model."

A. True
B. False

Answer: A

Example Codes

Group Project – Milestone 1 & 2

Take 10 minutes break…