Vector space modelling with ML

Stanislav Protasov

Refresh

- Term-document matrix
- Vector space model
- Distributive semantics
- Latent space
- LSA, PCA, SVD

Agenda

- LSA — what is missing
- Neural networks solving embedding task
  - word2vec, doc2vec
  - DSSM
  - BERT

LSA critics

Speed issue. Even optimized SVD is slow and requires memory and CPU time

- Fast Randomized SVD (facebook)
- Alternating Least Squares (ALS). Distributed and streaming versions

Model issue. PCA assumes **normal** data distribution, but life is complicated

- **pLSA**. Statistical independence (any distribution) vs linear orthogonality. Roots in statistics

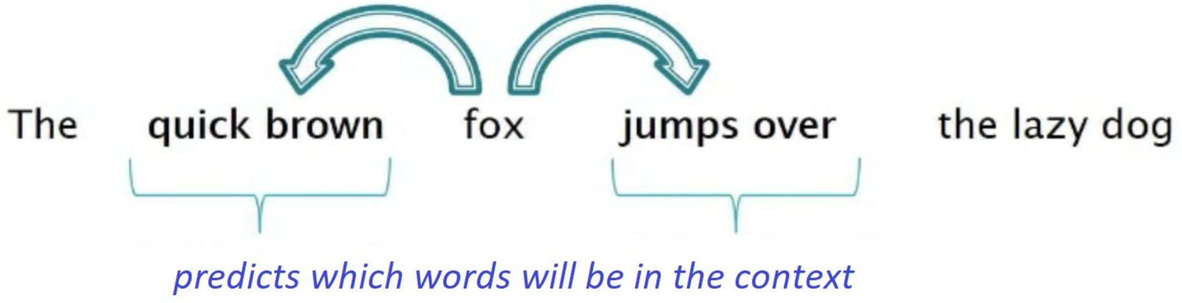
What about **adding new** words/texts?

Can we take some model and don’t care about distributions, statistics, memory and so on?

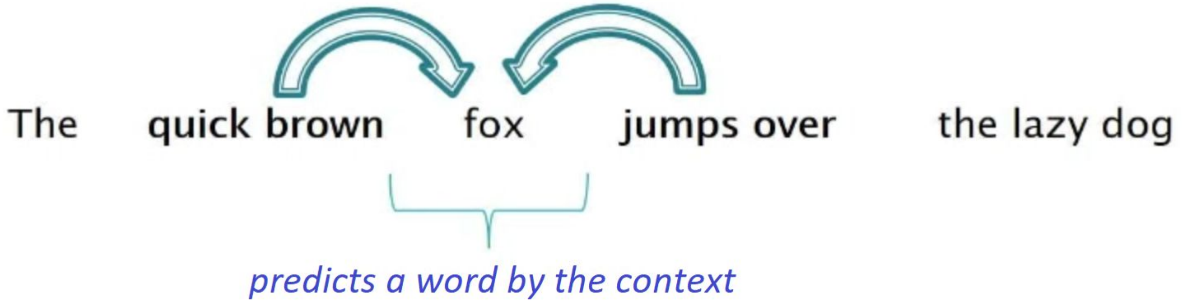
word2vec (2013) - group of methods

Do not compute, predict!

Skip-grams:



CBoW:



CBoW - Continuous bag of words

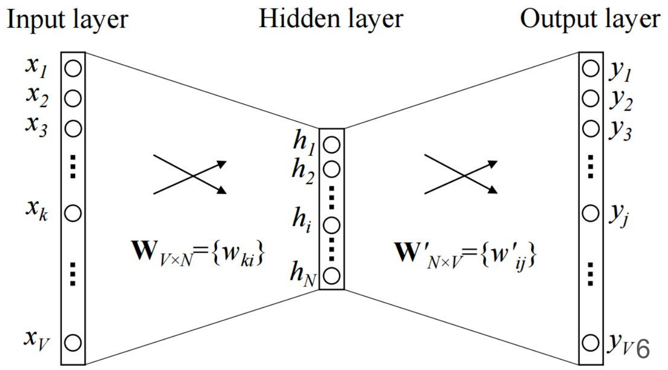
BoW - multiset of objects disregarding order (works for images, texts, …)

Input - one-hot **context** encoding (dict-size vector)

Output - overall **collection distribution** (dict size vector)

Activation (softmax) models a **likelihood** $p(w|c,\theta)$

Where is embedding?

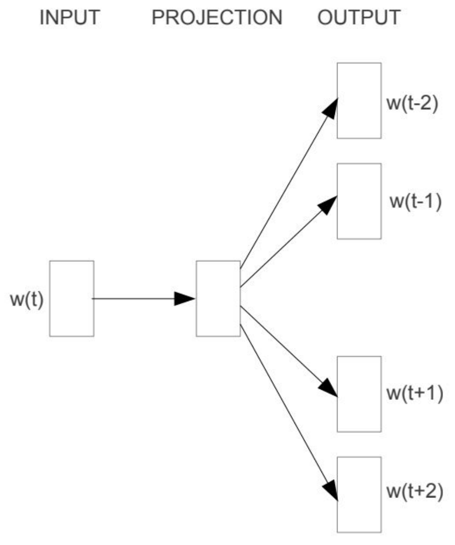


Continuous skip-gram model

“… instead of predicting the current word based on the context, it tries to **maximize classification of a word** based on **another word in the same** sentence” (paper)

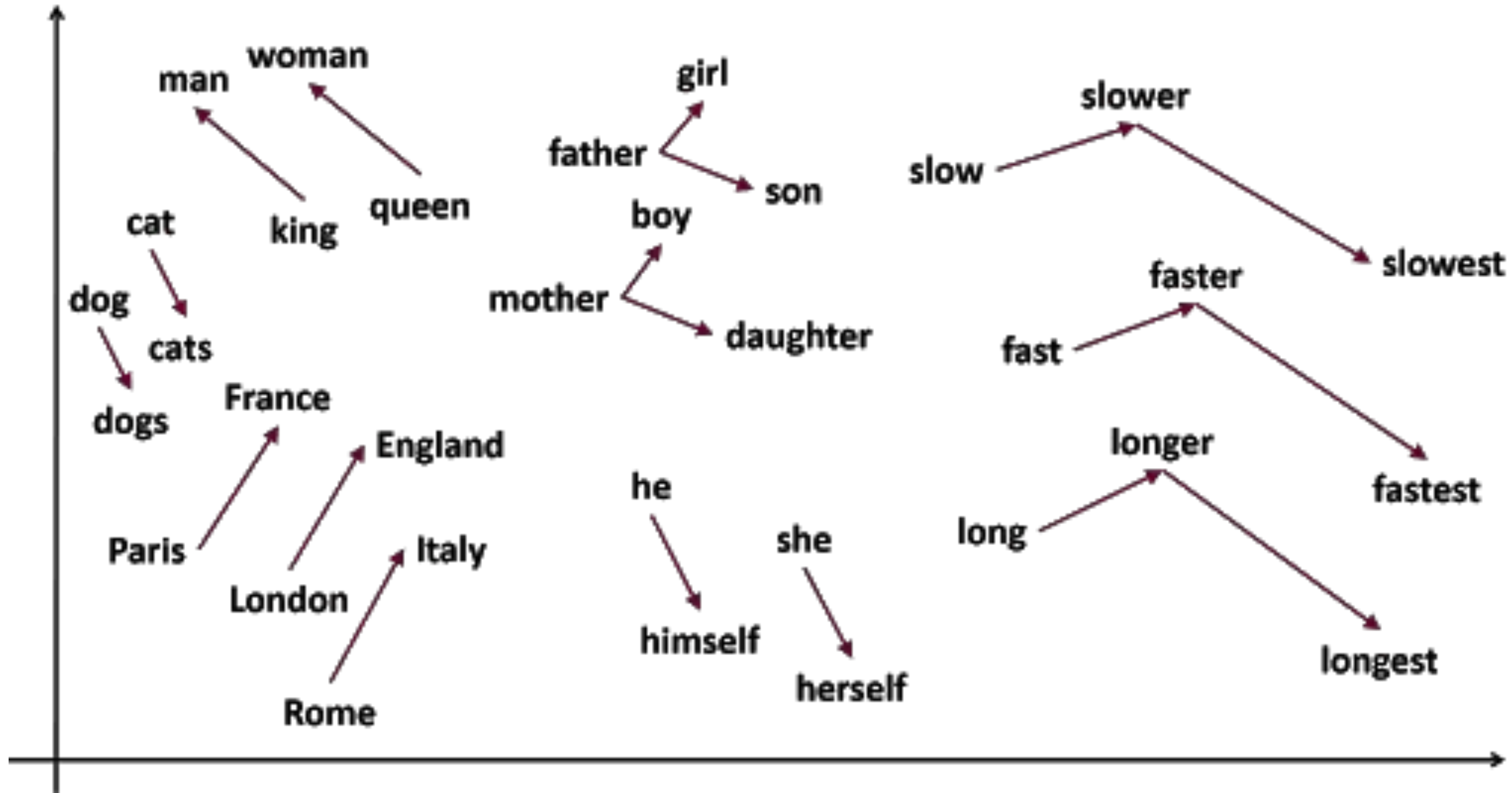
increasing the range improves quality of the resulting word vectors, but it also increases the computational complexity

...we give less weight to the **distant words** by **sampling less** from those words in our training examples.



Skip-gram

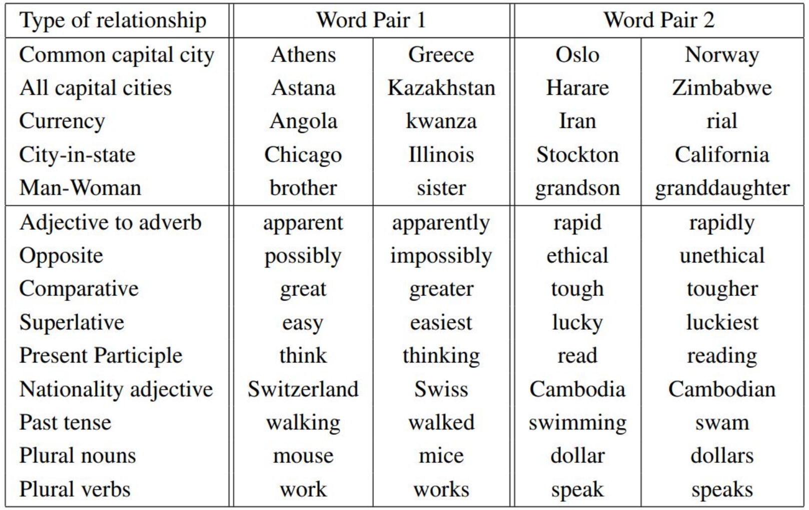
Bonus: vector space arithmetics



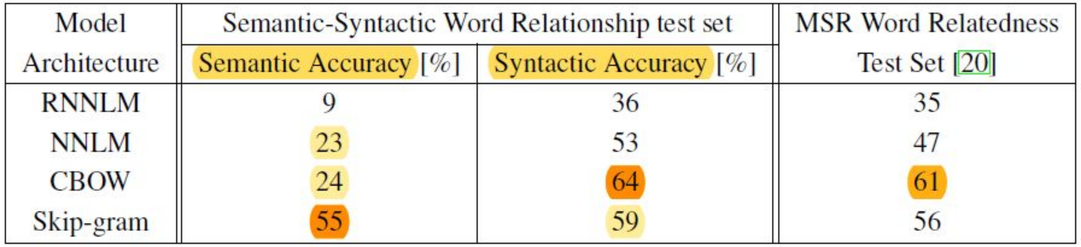
Training and quality

… we used three training epochs with **stochastic gradient descent** and **backpropagation**. We chose starting learning rate 0.025 and decreased it linearly, so that it approaches zero at the end of the last training epoch.

… there are 8869 **semantic** and 10675 **syntactic** questions.



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



Model	Semantic-Syntactic Word Relationship test set		MSR Word Relatedness
Architecture	Semantic Accuracy [%]	Syntactic Accuracy [%]	Test Set [20]
RNNLM	9	36	35
NNLM	23	53	47
CBOW	24	64	61
Skip-gram	55	59	56

word2vec (and almost everyone’s) problems

OOV - **out of vocabulary** or even underrepresented words.

- *today is approached with subword tokenization*

Grammar, *abbreviations*, *forms* and *homographs* — out of scope. (Paper don’t discuss lexers or stemmers)

Training depends on N (context size) ✕ D (dictionary).

Still works with **words**, **not paragraphs** or sentences.

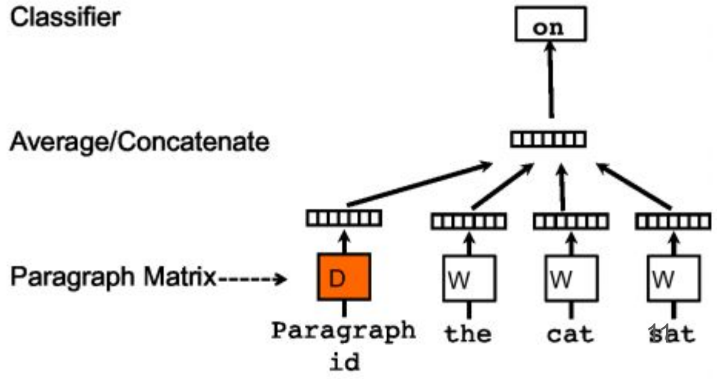
Doc2vec (Paragraph Vectors)

… we **concatenate the paragraph vector with several word vectors** from a paragraph and **predict the following word** in the given context. … paragraph vectors are unique among paragraphs, the word vectors are shared.

No syntax: using a parse tree to combine word vectors, has been shown to work for only sentences because it relies on parsing.

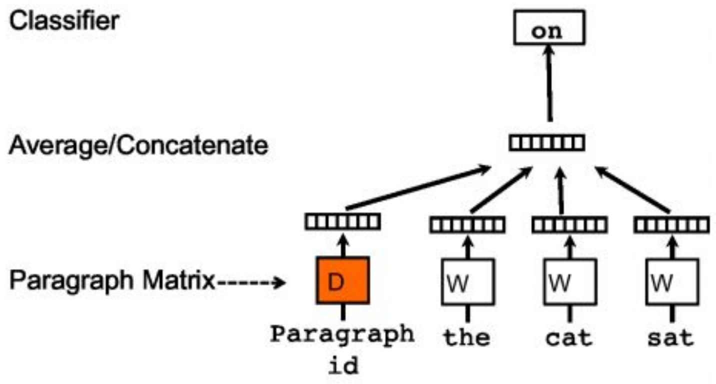
The **paragraph token** can be thought of as **another word.** Paragraph vector length can be different.

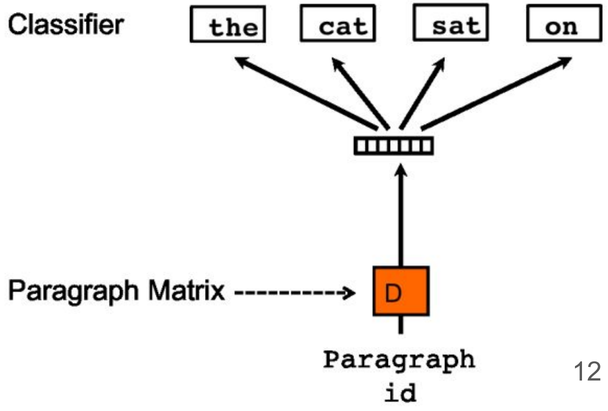
“the inference stage” to get paragraph vectors D for new paragraphs (never seen before) by adding more columns in D and gradient descending on D while holding W, U, b fixed



Details

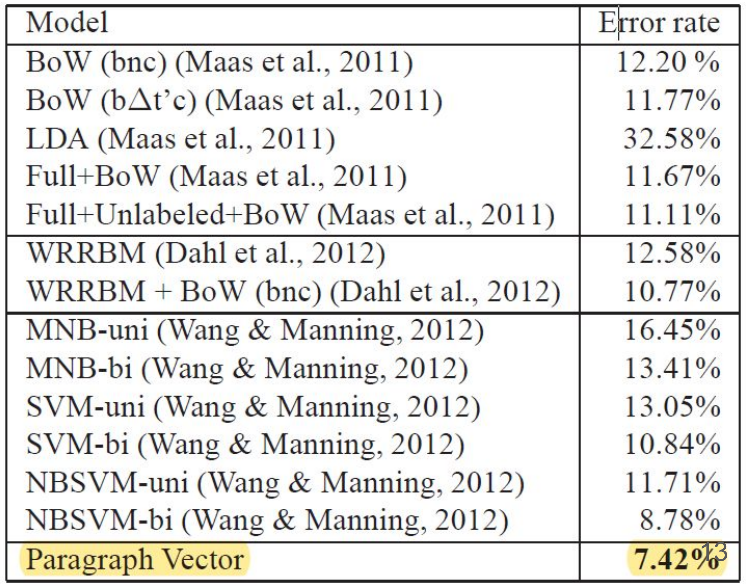
1. Paragraph Vector - **Distributed Memory** (PV-DM) - concatenate paragraph and word vectors
2. Paragraph Vector - **Distributed Bag of Words** (PV-DBOW) - ignore the context words in the input, but force the model to predict words randomly sampled from the paragraph in the output. Sample a text window, then sample a random word from the text window and form a classification task given the Paragraph Vector





Training and quality

- SGD
- NB Paragraph vector inference requires running GD!
- Sentiment analysis (5-class, 2-class) classification
  - Special characters such as **,.!?** are treated as a normal word



Model	Error rate
BoW (bnc) (Maas et al., 2011)	12.20 %
BoW (bΔt'c) (Maas et al., 2011)	11.77%
LDA (Maas et al., 2011)	32.58%
Full+BoW (Maas et al., 2011)	11.67%
Full+Unlabeled+BoW (Maas et al., 2011)	11.11%
WRRBM (Dahl et al., 2012)	12.58%
WRRBM + BoW (bnc) (Dahl et al., 2012)	10.77%
MNB-uni (Wang & Manning, 2012)	16.45%
MNB-bi (Wang & Manning, 2012)	13.41%
SVM-uni (Wang & Manning, 2012)	13.05%
SVM-bi (Wang & Manning, 2012)	10.84%
NBSVM-uni (Wang & Manning, 2012)	11.71%
NBSVM-bi (Wang & Manning, 2012)	8.78%
Paragraph Vector	**7.42%**

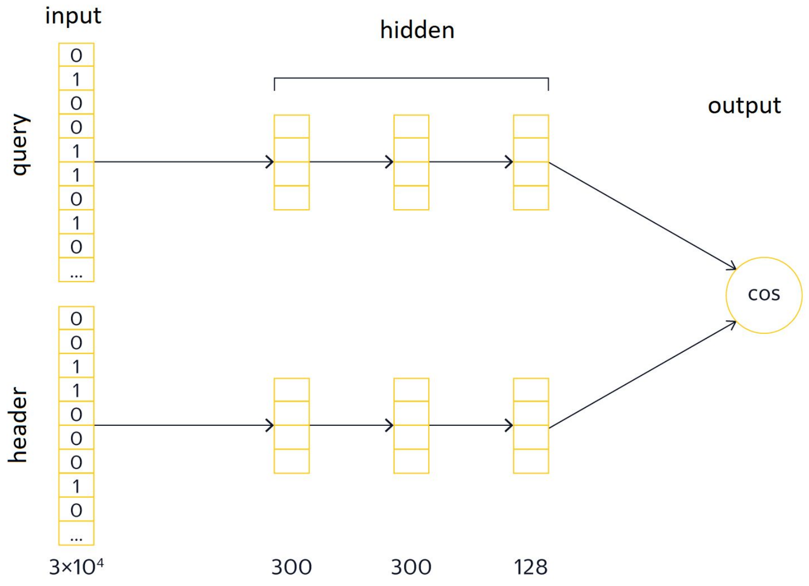
See also

GloVe - Global Vectors for Word Representation

Go deeper?

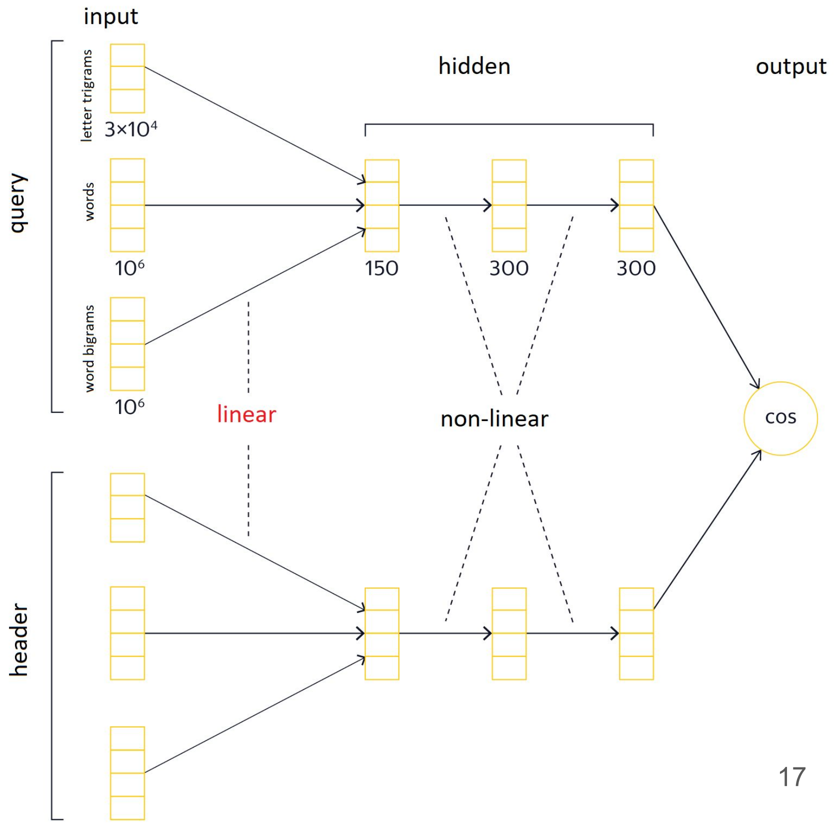
DSSM - Deep Structured Semantic Model (by MS)

- Original architecture:
  - Trained to predict cosine similarity
  - Uses bag of **letter trigrams**
- Important:
  - Initially created for **search** (uses specific metric)
- Problem:
  - Relatively **small input size** ($33^3/26^3$ of trigrams) for deep network
- Training:
  - Positive - clicked headers
  - Negative - shown but not clicked
  - **Not necessary relevance!**



DSSM by Yandex

- Input layer:
  - Trigrams
  - +1M of words
  - +1M of word bigrams
- Training:
  - Failed on random negatives
  - Failed on fake negatives
  - hard negative mining: take random best predictions and use as negative samples (similar to GANs, but simpler, as network fights itself)
  - Other target: dwelltime



See also

ELMo - Embeddings from Language Models (2-directional LSTM)

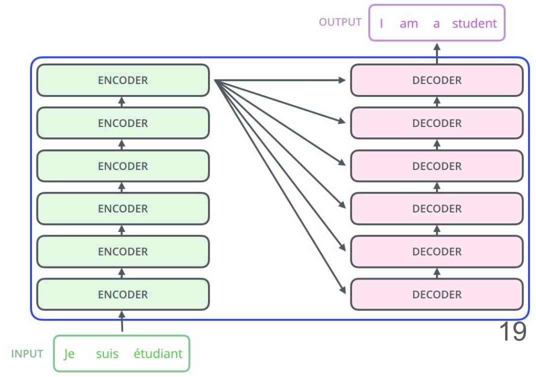
BERT (Bidirectional Encoder Representations from Transformers)

Created to learn **language model —** and to solve general language tasks

Modes:

- Trained to predict 15% of masked words
- Also trained to predict logical connections between phrases

“The major limitation is that standard language models are unidirectional, and this limits the choice of architectures that can be used during pre-training”



Reading

Papers and articles (links in presentation) on this topic