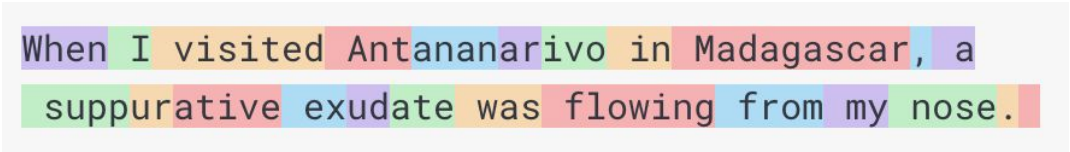
Vector search #2 – Text embeddings

Text

Sequence of tokens

- Query items
  - Keywords
  - question…
- Database items
  - Document
  - Paragraph
  - Sentence
  - Passage…
- Tokens → units than can be enumerated 1..W
  - Convenient: words
  - n-grams – overlapping or not
  - Tokenization



https://platform.openai.com/tokenizer

Word statistics

- 10k - 1M
  - Vocabulary of the language
  - Named entities
- Normalization (stemming)
  - Normalize upper/lower case
  - Remove plural
  - Remove conjugations
  - Remove declension
  - Split words for morphologically rich languages

Kaasschaafverkoper

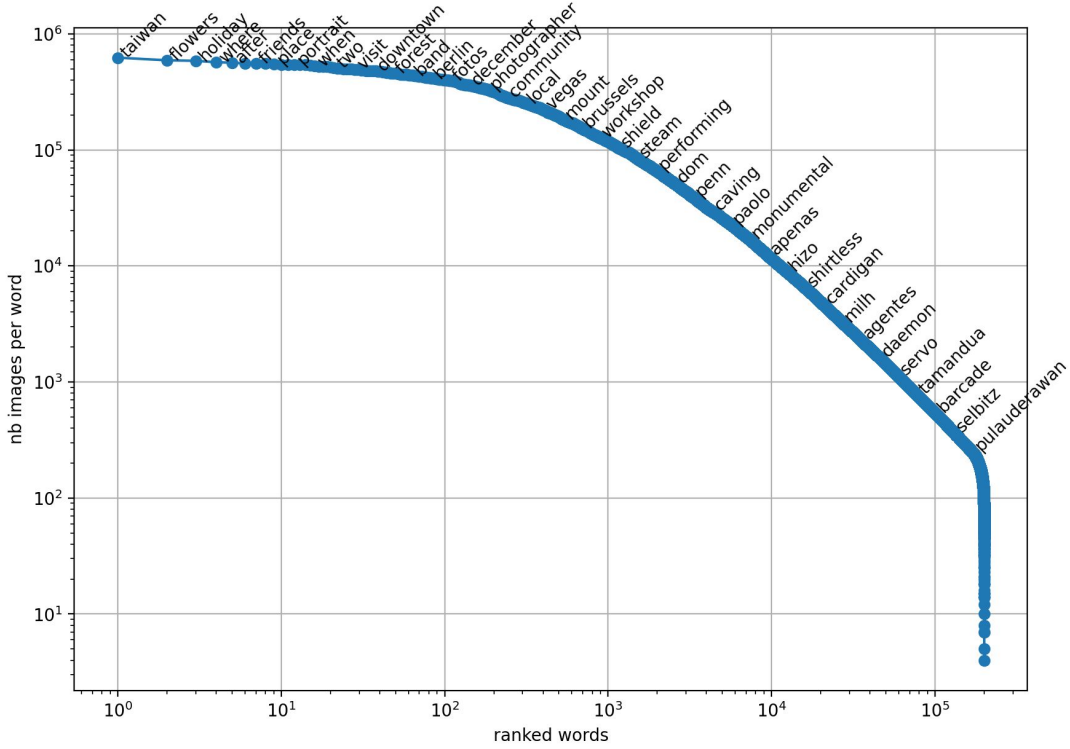
Cheese Slicer Salesperson

Very unbalanced

- Common:
  - A, the, it, etc.

- Less common:

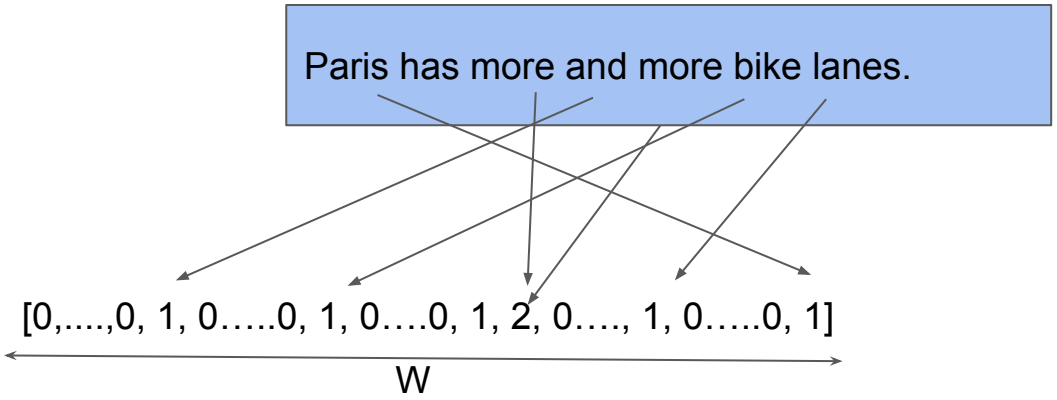
60026	exudative	j
60027	shakti	n
60028	shearling	j
60029	sheerly	r
60030	short-selling	n
60031	phytic	j
60032	piedmontese	j



Bag of words

Bag of words

- The set of words in a document
- Discard order: “bag”
  - Keep counts
  - Word → number in 1..W



- Sparse vector of size W (6 non-zeros / 10k)

Match words between query and database docs

I ride a bike in Paris.

Paris has more and more bike lanes.

Bike rides in Amsterdam

- Inverted index
  - Map word → list of documents that contain the word
  - “Inverted list”
  - Word stemming (remove plural)
  - Stop words (common words)

a	1
Amsterdam	3
and	2
bike	1, 2, 3
has	2
in	1, 3
lane	2
more	2
Paris	1, 2
ride	1,3

TF-IDF weighting

- How important is a term (word) to a document?
- TF: term frequency
  - More important if the term appears more in the doc
- IDF: inverse document frequency
  - More important if the term is more rare in the corpus D
  - For common terms → near 0
  - Justifies stop words

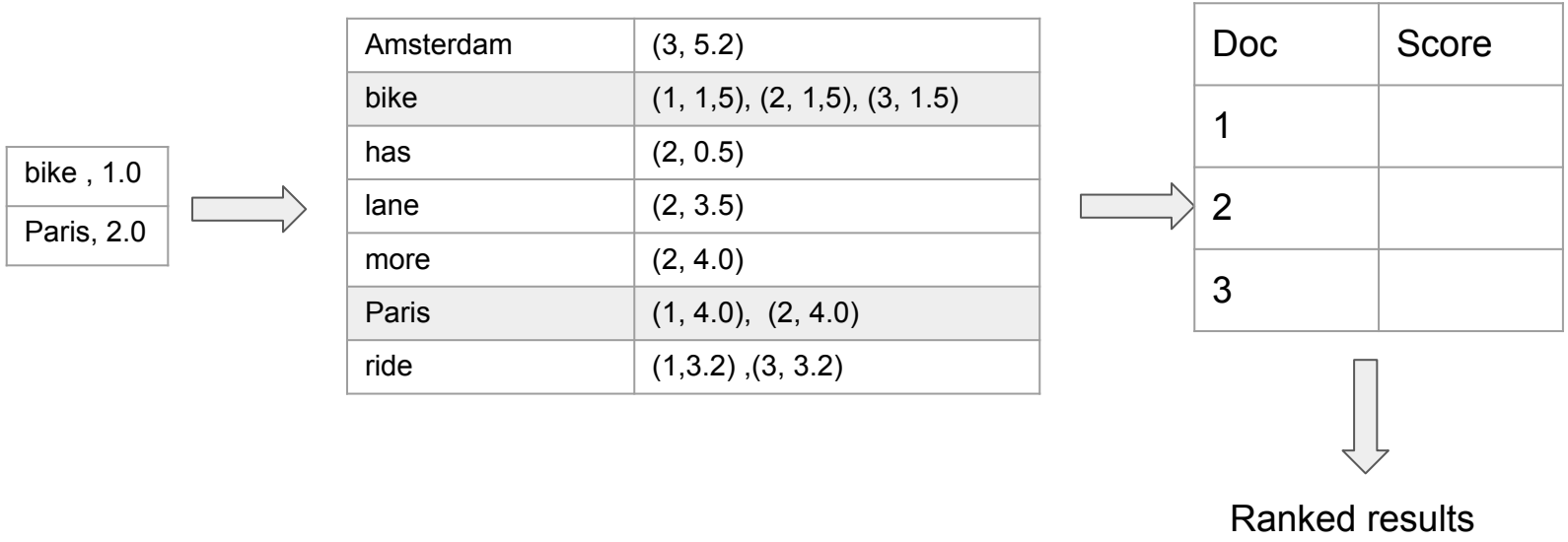
$$\mathrm{tf}(t,d) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}},$$

$$\mathrm{idf}(t,D) = \log \frac{N}{|\{d \in D : t \in d\}|}$$

$$\mathrm{tfidf}(t,d,D) = \mathrm{tf}(t,d) \cdot \mathrm{idf}(t,D)$$

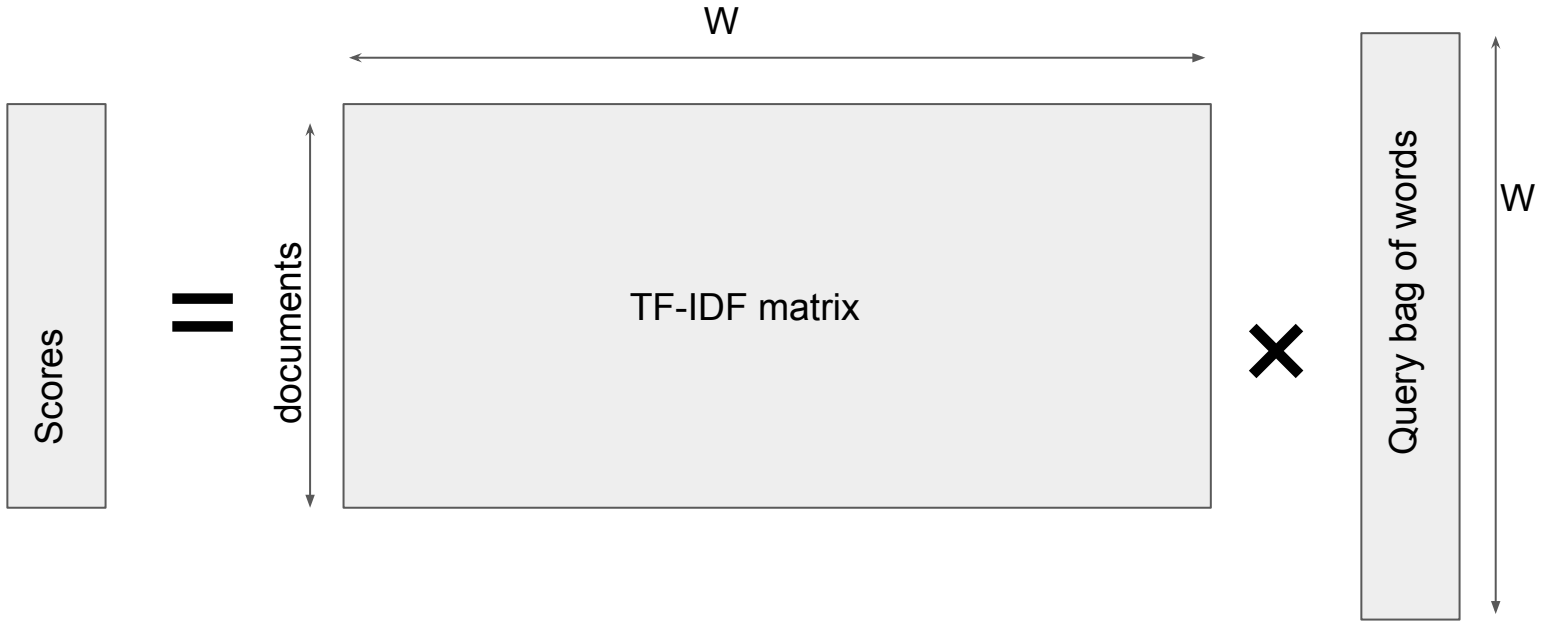
Search with TF-IDF weighting

- TF-IDF weights in inverted list entries
- Sum query term entries over documents to get scores



Equivalence with sparse matrix-vector product

- Inverted file = CSR
  - Compressed Sparse Row representation
- Efficient implmentation (see scipy.sparse)



BM25 ranking

- Widely used variant of TF-IDF
  - Depends on scalar parameters

$$\text{score}(D, Q) = \sum_{i=1}^{n} \text{IDF}(q_i) \cdot \frac{f(q_i, D) \cdot (k_1 + 1)}{f(q_i, D) + k_1 \cdot \left(1 - b + b \cdot \frac{|D|}{\text{avgdl}}\right)}$$

$$\text{IDF}(q_i) = \ln\left(\frac{N - n(q_i) + 0.5}{n(q_i) + 0.5} + 1\right)$$

About word matching

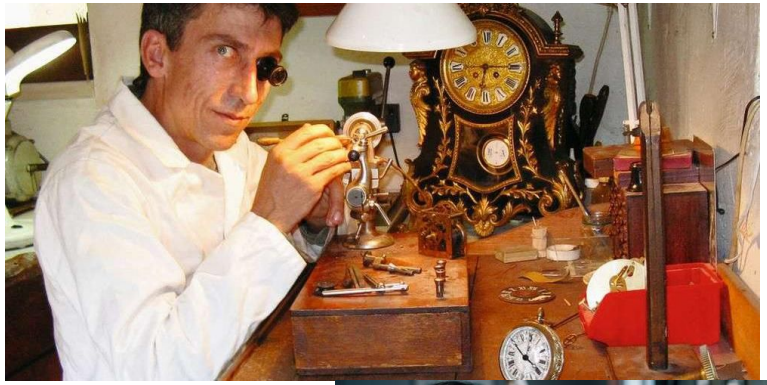
- Good performance
  - Fast
  - Excellent for keyword queries “python io module”
  - BOW = sum of individual word embeddings

- Limitation
  - Exact matching
  - “Tall” is as different from “large” as “small”

- Unable to take semantics into account
  - Purely statistical

Embedding discrete items

Machine learning preliminaries

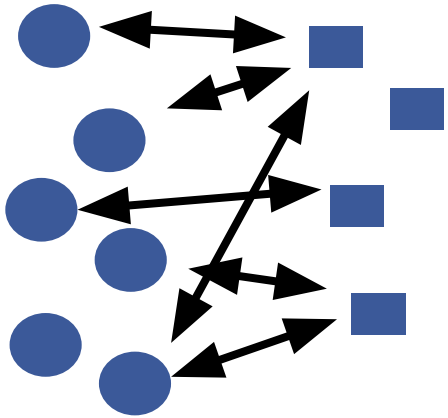


- Subtle craftsmanship
  - little guidance from theory 😕
  - Many methods don’t work unless parameters are set properly – trial and error – sweeps
  - details in the papers are important!
- Train a function y = f(x, theta)
  - Given a training dataset {x_i, y_i}
  - Iterative optimization of theta

- Interaction between 4 components
  - A function that is “sufficiently complex” for the task: the neural net
  - A loss function that reflects the task
  - A way to present the dataset
  - An optimization schedule: how to update the theta ?

Creating embeddings for discrete items

- A set of items
  - We don’t have/use a priori information (Fields, properties, etc.)
  - We need embeddings for them in some dimension d
- Relations between items
  - Sparse observations
  - Scores
- Find embeddings
  - such that similarities explain the relations
  - Nearby embeddings → high scores
  - Far away embeddings → low scores



A step aside: recommender systems

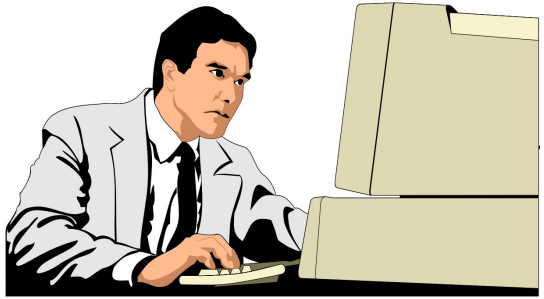
- N users
  - Customers

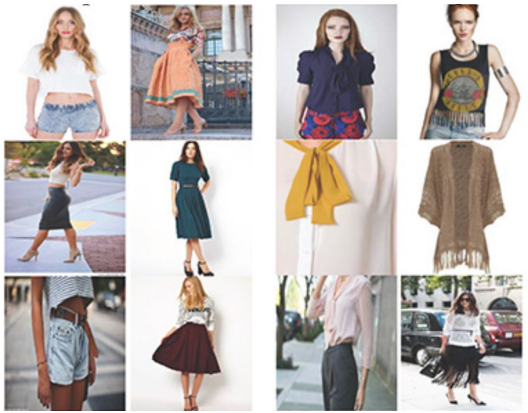
$$\{u_1, ..., u_N\} \in \mathbb{R}^d$$

- M items
  - Products, movies, etc.

$$\{v_1, ..., v_M\} \in \mathbb{R}^d$$

- Score for user i, item j: $s_{i,j} \in \mathbb{R}$
  - Direct: stars, grades, …
  - Indirect: previous purchases, clicks,...
- Regularity:
  - similar users tend to choose similar items
  - “Collaborative filtering”





Recommender systems – optimization

- Reproduce scores with dot product between user and item

$$s_{i,j} \approx \langle u_i, v_j \rangle$$

- Predict non-observed scores
- Formalization:
  - Objective = minimize the loss
  - Linear algebra works best with quadratic loss…

$$\text{Loss} = \sum_{(i,j) \in S} \left(s_{i,j} - \langle u_i, v_j \rangle\right)^2$$

Set of obseved scores

Recommender systems – all scores known

- If the full score matrix was known

$$S = \{1, ..., N\} \times \{1, ..., M\}$$

- Matrix formulation

$$U = [u_1, ..., u_N], V = [v_1, ..., v_M], R = [s_{ij}] \in \mathbb{R}^{N \times M}$$

$$\text{Loss} = \|UV^\top - R\|_{\text{F}}^2$$

- Low-rank approximation of R
  - Matrix factorization
  - SVD, see Edo's class

Recommender systems – iterative solution

- R is not full…
  - otherwise what's there to recommend?

$$\text{Loss} = \sum_{(i,j)\in S} \left(s_{i,j} - \langle u_i, v_j \rangle\right)^2$$

$$\text{Loss} = \sum_{x\in S} f(x,\theta)$$

$$\theta = \{u_i, ..., u_N, v_1, ..., v_M\}$$

- Stochastic gradient descent
- Random initialization of parameters
- Iterate
  - Pick a random sample x of training set S
  - Update the parameters in the direction of the gradient
  - Learning rate

$$\theta_{t+1} = \theta_t - \lambda \frac{\partial f(x,\theta)}{\partial \theta}$$

Exercise:

- Compute the gradient $\dfrac{\partial f(x, \theta)}{\partial \theta}$

- Complexity of a gradient step?

End side step

- Embeddings for a set of items can be built from relationships

- Usually not as simple
  - Need regularization term
  - Normalization
  - → Integrates smoothly in iterative algorithm
  - Sampling strategies –

- Tradeoff on dimensionality d:
  - If d is too small → insufficient capacity, not accurate
  - If d is too large → overfitting – unable to generalize

Word embeddings

Word2vec

- Trained on corpus of documents
  - No additional information (supervision)
  - Trained on docs with 1B words total

- Pull together words that are nearby in document text

- Skip-gram model
  - “Predict” a missing word from context
  - Probabilistic formulation (not really used)

[Mikolov & al. Distributed representations of words and phrases and their compositionality. NIPS’13]

Skip gram model

- Current word representation: u_w
- Neighboring word representations: v_w
  - Neighborhood size: c=3

For instance, any given movie can, to a rough degree of approximation, be described in terms of some basic attributes such as overall quality, whether it's an action **movie** or a comedy, what stars are in it, and so on. And every user's preferences can likewise be roughly described in terms of whether they tend to rate high or low, whether they prefer action movies or comedies, what stars they like, and so on.

- Pull together u_*movie* and v_*action*, v_*or*, v_*a*, v_*comedy*
- Then move on to the next word (“or”)

https://sifter.org/~simon/journal/20061211.html

Basic formulation

whether it's an action **movie** or a comedy, what stars

w_{t-3} w_{t-2} w_{t-1} w_t w_{t+1} w_{t+2} w_{t+3}

- Compare embeddings with dot product
- Use “SGD friendly” optimization objective
  - With softmax

$$\text{Loss} = -\sum_{t=1}^{T} \sum_{j \in \{-c,..,-1,1,...,c\}} \log \frac{\exp\langle u_{w_t}, v_{w_{t+j}} \rangle}{\sum_{w=1}^{W} \exp\langle u_{w_t}, v_w \rangle}$$

Sum over doc words

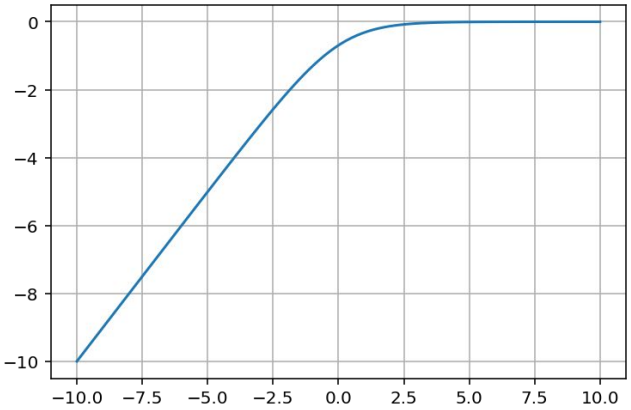
Sum over word neighborhood

Score(t, j)

Softmax w.r.t. all words of the vocabulary

Practical implementation

- Bottleneck is the softmax computation
  - Complexity O(d * W)
  - Can be approximated with hierarchical softmax
- Negative sampling
  - Replace softmax with log of sigmoid
  - Encourage large dot product but not too much
  - And a sample of k “noise” words N(t, j)



$$\log \sigma(x) = \log(1/(1+e^{-x}))$$

$$\text{Score}(t,j) = \log(\sigma\langle u_{w_t}, v_{w_{t+j}}\rangle) - \sum_{w \in N(t,j)} \log(\sigma\langle u_{w_t}, v_w\rangle)$$

- Subtle choice of sampling
  - Based on word frequencies

Both NCE and NEG have the noise distribution $P_n(w)$ as a free parameter. We investigated a number of choices for $P_n(w)$ and found that the unigram distribution $U(w)$ raised to the $3/4$rd power (i.e., $U(w)^{3/4}/Z$) outperformed significantly the unigram and the uniform distributions, for both NCE and NEG on every task we tried including language modeling (not reported here).

Evaluation of word embeddings: nearest word

- Nearest words
  - Limited capability…

Model (training time)	Redmond	Havel	ninjutsu	graffiti	capitulate
Collobert (50d) (2 months)	conyers lubbock keene	plauen dzerzhinsky osterreich	reiki kohona karate	cheesecake gossip dioramas	abdicate accede rearm
Turian (200d) (few weeks)	McCarthy Alston Cousins	Jewell Arzu Ovitz	- - -	gunfire emotion impunity	- - -
Mnih (100d) (7 days)	Podhurst Harlang Agarwal	Pontiff Pinochet Rodionov	- - -	anaesthetics monkeys Jews	Mavericks planning hesitated
Skip-Phrase (1000d, 1 day)	Redmond Wash. Redmond Washington Microsoft	Vaclav Havel president Vaclav Havel Velvet Revolution	ninja martial arts swordsmanship	spray paint grafitti taggers	capitulation capitulated capitulating

Table 6: Examples of the closest tokens given various well known models and the Skip-gram model trained on phrases using over 30 billion training words. An empty cell means that the word was not in the vocabulary.

← results for word2vec extended to pair of words

[Mikolov & al. Efficient Estimation of Word Representations in Vector Space, NIPS'13]

Evaluation of word embeddings

- Specially designed task: word relationship test

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

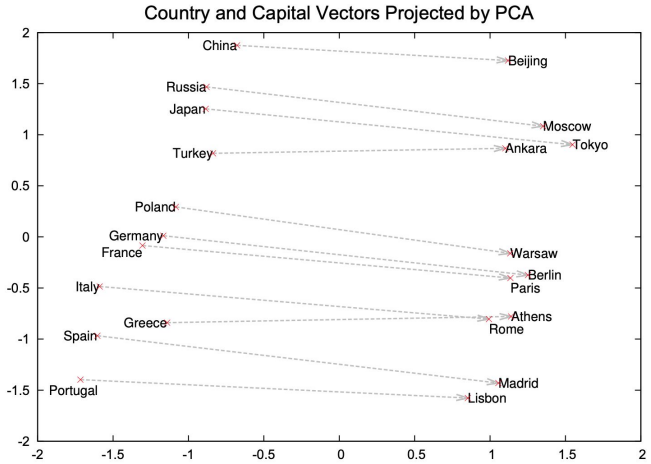
Semantic (8869)

Syntactic (10675)

Word arithmetic

Somewhat surprisingly, these questions can be answered by performing simple algebraic operations with the vector representation of words. To find a word that is similar to *small* in the same sense as *biggest* is similar to *big*, we can simply compute vector $X = vector(\text{"}biggest\text{"}) - vector(\text{"}big\text{"}) + vector(\text{"}small\text{"})$. Then, we search in the vector space for the word closest to X measured by cosine distance, and use it as the answer to the question

- The country-to-capital vector:



- Dependence on dim vectors:

Dimensionality / Training words	24M	49M	98M	196M	391M	783M
50	13.4	15.7	18.6	19.1	22.5	23.2
100	19.4	23.1	27.8	28.7	33.4	32.2
300	23.2	29.2	35.3	38.6	43.7	45.9
600	24.0	30.1	36.5	40.8	46.6	50.4

[Bojanowski & al. Enriching Word Vectors with Subword information, ACL'17]

Word embeddings with sub-words (fasttext)

- Collect all n-gram sub-words for a word

tiling

3	<ti	til	ili	lin	ing	ng>
4	<til	tili	ilin	ling	ing>	
5	<tili	tilin	iling	ling>		
6	<tilin	tiling	iling>			

- Sum up n-gram embeddings + word embedding
  - Hashed n-gram → limited to 2M embeddings
  - Collisions unlikely…
  - Replace sub-word embedding in skip-gram model

Results on fastText

- Significant improvement on syntactic analogy
- Needs less training data
- Works better on morphologically rich languages

		sg	cbow	sisg
CS	Semantic	25.7	27.6	27.5
	Syntactic	52.8	55.0	77.8
DE	Semantic	66.5	66.8	62.3
	Syntactic	44.5	45.0	56.4
EN	Semantic	78.5	78.2	77.8
	Syntactic	70.1	69.9	74.9
IT	Semantic	52.3	54.7	52.3
	Syntactic	51.5	51.8	62.7

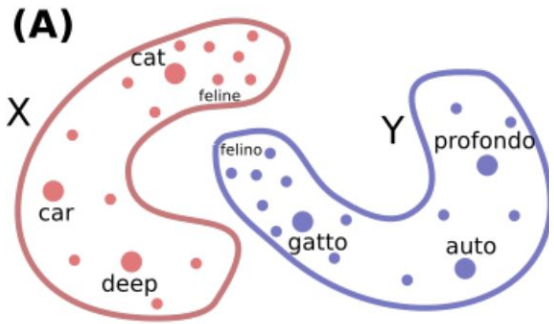
query	tiling	tech-rich	english-born	micromanaging	eateries	dendritic
sisg	tile flooring	tech-dominated tech-heavy	british-born polish-born	micromanage micromanaged	restaurants eaterie	dendrite dendrites
sg	bookcases built-ins	technology-heavy .ixic	most-capped ex-scotland	defang internalise	restaurants delis	epithelial p53

Table 7: Nearest neighbors of rare words using our representations and skipgram. These hand picked examples are for illustration.

Word translation

[Word translation without parallel data, Conneau et al, ICLR'18]

- Analogy task suggests there is a way to translate
  - Given translation of anchor words



- Maybe a rigid structure in word embeddings?

- Could we align them ?
  - → effortless translation
  - W unitary matrix

$$y_{\text{it}} \approx W x_{\text{en}}$$

Estimating W

- If there are a few word (<<W) correspondences

$$\begin{aligned} X &= [x_{\text{en}}^1 \quad x_{\text{en}}^2 \quad \cdots \quad x_{\text{en}}^n] \\ Y &= [y_{\text{it}}^1 \quad y_{\text{it}}^2 \quad \cdots \quad y_{\text{it}}^n \ ] \end{aligned}$$

- Then W can be estimated with
  - Closed form solution based on SVD

$$W^\star = \underset{W \in M_d(\mathbb{R})}{\operatorname{argmin}} \|WX - Y\|_{\text{F}}$$

- Translation

$$t = \operatorname{argmax}_t \cos(Wx_s, y_t)$$

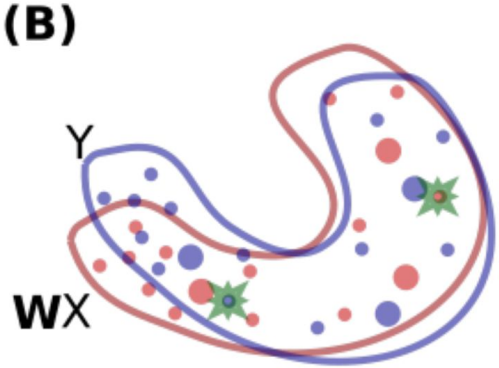
What if we don’t have initial words?

- Use adversarial training
- Discriminator function D(x, theta)
  - Higher for source language (x)
  - Lower for target language (y)
  - Parameters theta
- Batches of (Wx, Wx, y, Wx, y, Wx, Wx)
- Step 1: optimize discriminator

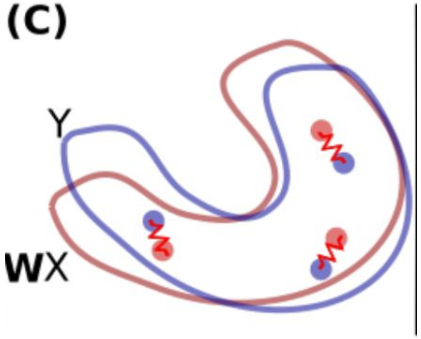
$$\text{Loss}(\theta) = -\sum_i D(Wx_i, \theta) + \sum_j D(y_j, \theta)$$

- Step 2: optimize W to fool the discriminator

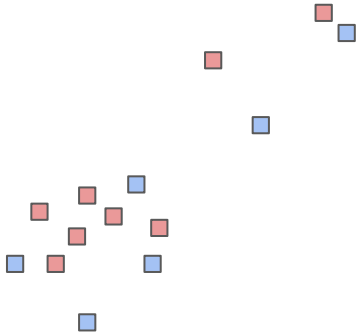
$$\text{Loss}(W) = \sum_i D(Wx_i, \theta) - \sum_j D(y_j, \theta)$$



Refinement with reliable word matches



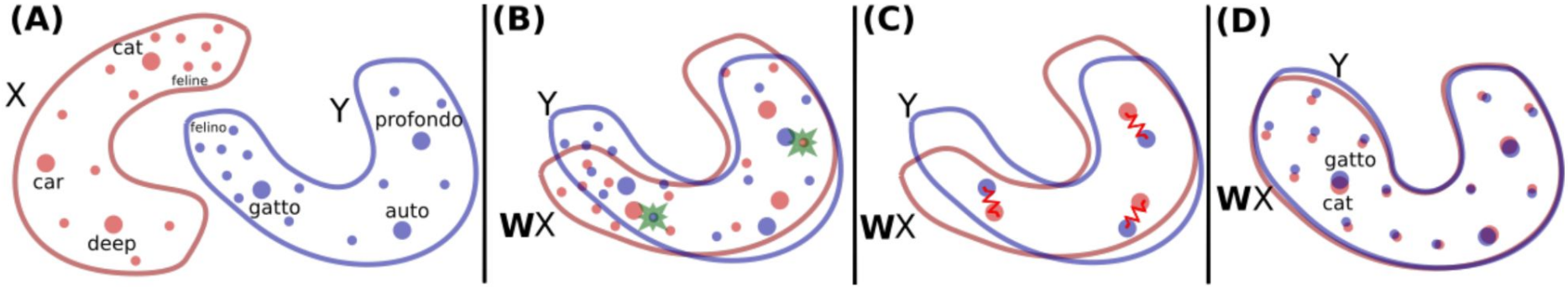
- Select a subset of word pairs and re-estimate W
- How to select reliable pairs?
  - General method to select a subset of pairs
- k-NN search + additional criteria
- Reverse nearest neighbors
  - Wx must be a nearest neighbor of y
  - y must be a nearest neighbor of x
- CSLS criterion
  - Contrast with similarity to other neighbors



$$\mathrm{CSLS}(Wx_s, y_t) = 2\cos(Wx_s, y_t) - r_{\mathrm{T}}(Wx_s) - r_{\mathrm{S}}(y_t)$$

$$r_{\mathrm{T}}(Wx_s) = \frac{1}{K} \sum_{y_t \in \mathcal{N}_{\mathrm{T}}(Wx_s)} \cos(Wx_s, y_t)$$

Results



	en-es	es-en	en-fr	fr-en	en-de	de-en	en-ru	ru-en	en-zh	zh-en	en-eo	eo-en
Methods with cross-lingual supervision and fastText embeddings												
Procrustes - NN	77.4	77.3	74.9	76.1	68.4	67.7	47.0	58.2	40.6	30.2	22.1	20.4
Procrustes - ISF	81.1	82.6	81.1	81.3	71.1	71.5	49.5	63.8	35.7	**37.5**	29.0	27.9
Procrustes - CSLS	81.4	82.9	81.1	**82.4**	73.5	**72.4**	**51.7**	**63.7**	**42.7**	36.7	**29.3**	25.3
Methods without cross-lingual supervision and fastText embeddings												
Adv - NN	69.8	71.3	70.4	61.9	63.1	59.6	29.1	41.5	18.5	22.3	13.5	12.1
Adv - CSLS	75.7	79.7	77.8	71.2	70.1	66.4	37.2	48.1	23.4	28.3	18.6	16.6
Adv - Refine - NN	79.1	78.1	78.1	78.2	71.3	69.6	37.3	54.3	30.9	21.9	20.7	20.6
Adv - Refine - CSLS	**81.7**	**83.3**	**82.3**	82.1	**74.0**	72.2	44.0	59.1	32.5	31.4	28.2	**25.6**

Table 1: Word translation retrieval P@1 for our released vocabularies in various language pairs. We

Conclusion on word embeddings

- Rich semantic information
- Very simple to train
  - Outperforms multi-layer models
  - Needs lots of training data
- Emerging arithmetic properties
- Basis for the following
  - Input to more complex language models

Document embeddings

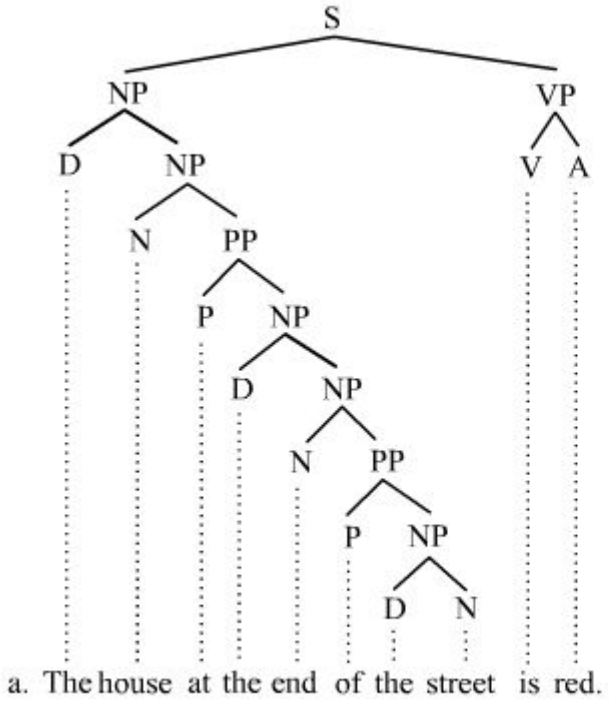
Multi-word embeddings

- Sequence of words
  - Whole documents
  - Sentence
  - phrase
  - Paragraph / passage
- Combinatorial size
  - Possible for 2-word units

Czech + currency	Vietnam + capital	German + airlines	Russian + river	French + actress
koruna	Hanoi	airline Lufthansa	Moscow	Juliette Binoche
Check crown	Ho Chi Minh City	carrier Lufthansa	Volga River	Vanessa Paradis
Polish zolty	Viet Nam	flag carrier Lufthansa	upriver	Charlotte Gainsbourg
CTK	Vietnamese	Lufthansa	Russia	Cecile De

Table 5: Vector compositionality using element-wise addition. Four closest tokens to the sum of two vectors are shown, using the best Skip-gram model.

  - Cannot build a table of all document → embedding
- What to do with embeddings?
  - Nearest neighbor search (Translation)
  - Question answering (context)



NP noun phrase
PP preposition
VP vector phrase

Word pooling

[Offline bilingual word vectors, orthogonal transformations and the inverted softmax, Smith et al, ICLR'17]

- Sum of word vectors
  - Continuous bag of words (CBOW, not to be confused with the word2vec model)
  - Possibly with IVF weighting (increase weight of less frequent words)

- Works for sentence translation retrieval
  - 2000 query sentences
  - 200k targets that contain a translation of the sources
  - Sentence embedding with words

	English to Italian			Italian to English		
	P@1	P@5	P@10	P@1	P@5	P@10
Methods with cross-lingual supervision						
Mikolov et al. (2013b) †	10.5	18.7	22.8	12.0	22.1	26.7
Dinu et al. (2015) †	45.3	72.4	80.7	48.9	71.3	78.3
Smith et al. (2017) †	54.6	72.7	78.2	42.9	62.2	69.2
Procrustes - NN	42.6	54.7	59.0	53.5	65.5	69.5
Procrustes - CSLS	**66.1**	77.1	80.7	**69.5**	**79.6**	**83.5**
Methods without cross-lingual supervision						
Adv - CSLS	42.5	57.6	63.6	47.0	62.1	67.8
Adv - Refine - CSLS	65.9	**79.7**	**83.1**	69.0	**79.7**	83.1

Vector sum pooling exercise

[A group testing framework for similarity search in high-dimensional spaces, Shi et al, ACM MM'14]

- Given
  - Set of n vectors uniform on sphere dimension d
  - Unit query vector q
- Compute
  - If all vectors are independent of q
  - If q is one of the vectors
- → a general property of pooling

$$X = \sum_{i=1}^{n} X_i$$

$$E[\langle X, q \rangle] = ?$$

$$\mathrm{Var}[\langle X, q \rangle] = ?$$

Sentence encoders

A language model

[Improving Language Understanding by Generative Pre-Training, Radford et al, OpenAI tech report ’18]

- Probabilistic formulation
  - Sequence of tokens
  - Predict next token’s likelihood given a context window of size k

$$\mathcal{U} = \{u_1, \ldots, u_n\}$$

$$L_1(\mathcal{U}) = \sum_i \log P(\underbrace{u_i | u_{i-k}, \ldots, u_{i-1}}; \Theta)$$

Given by a neural net with parameters Theta

- At inference: does this belong to the language?
- Metric: perplexity
  - Lower = better

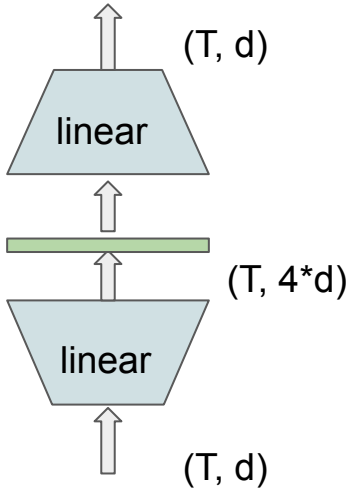
$$PPL = \exp\left(-\frac{1}{n} L_1(\mathcal{U})\right)$$

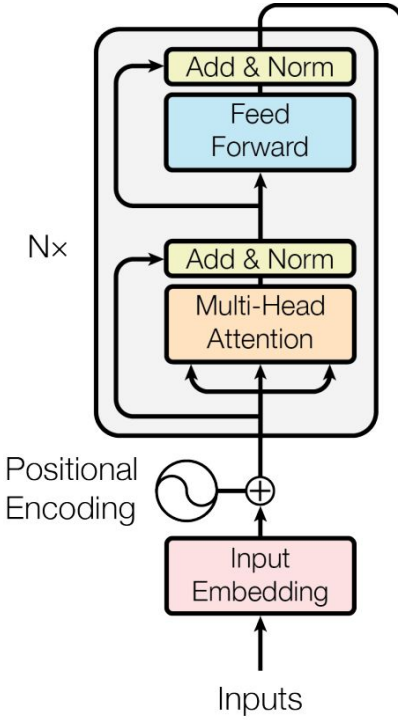
- Unsupervised
  - Can be trained on any corpus of texts
  - Larger is better…

[Attention is all you need, Vaswani et al, NIPS'17]

Language models based on transformers

- Initially applied to sequence-to-sequence modeling (translation)
  - Encoder + decoder, with supervision
- Input = sequence of token embeddings
  - Token → look-up table
  - positional encoding
  - Arbitrary length
- Sequence of Blocks
  - Attention
  - Feed forward
  - Applied as residuals



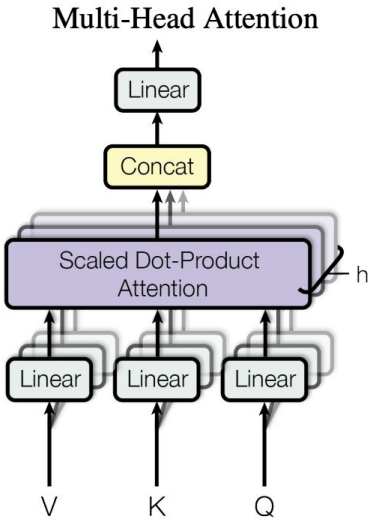


The attention function

- Pass input through 3 linear layers → Q, K, V
- Attention mechanism
  - Interaction between all vectors of the sequence
  - Row-wise softmax
  - Weighted sum of V matrix rows

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

- Multi-head
  - Vector decomposed into sub-vectors
  - Processed independently
  - Concatenated back in the end



What the attention is attending on

[Survey on Self-Supervised Multimodal Representation Learning and Foundation Models, Thapa, arxiv'22]

- Hard to visualize
  - Multiple layers
  - Multiple heads…

- LARGE language models
  - Add more layers 12 → 80
  - Increase embedding dimensionality 768 →1024
  - Increase number of heads 12 → 16
  - … and increase context size!

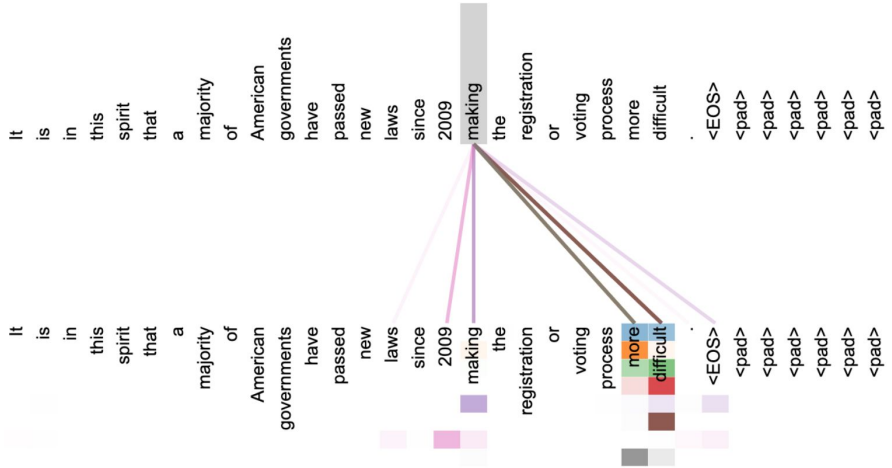


Figure 3: An example of the self-attention mechanism following long-distance dependencies in the encoder self-attention. Many of the attention heads attend to a distant dependency of the verb 'making', completing the phrase 'making...more difficult' [Vaswani et al., 2017].

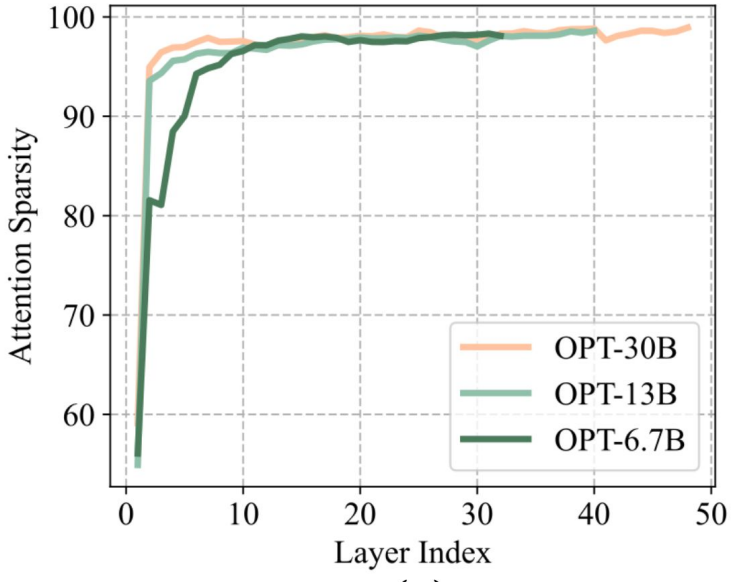
Scaling up the context size

[H2O: Heavy-Hitter Oracle for Efficient Generative Inference of Large Language Models, Zhang et al, Arxiv'23]

- More context:
  - More information for output
  - Up to 49k tokens
- Quadratic cost of Q * K
- Exploit attention sparsity
  - More sparse for upper layers
  - Find non-0 entries is max inner product search

$$\text{softmax}(\frac{QK^T}{\sqrt{d_k}})$$

- External memory…
  - Later in this class



BERT

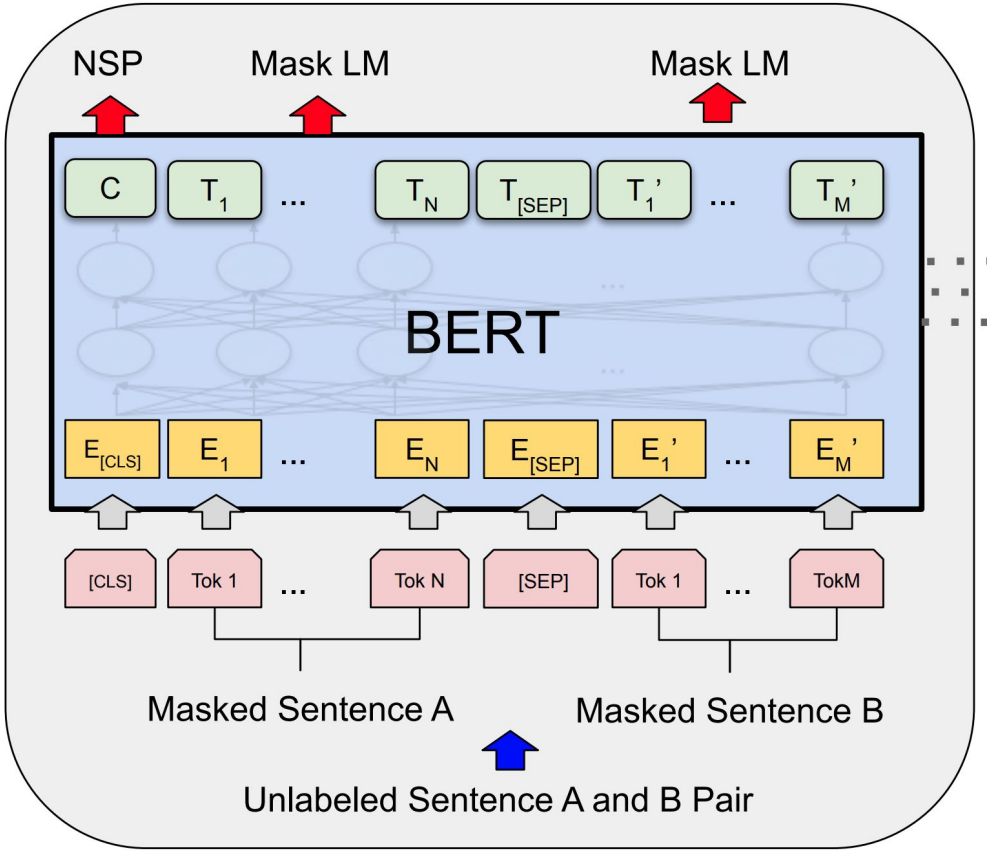
[Bert: Pre-training of deep bidirectional transformers for language understanding, Devlin et al, ArXiV'19]

- Simpler use case than sequence to sequence
  - Just interested in embedding for one sequence

- Unsupervised pre-training
  - Large corpus
  - Language model-like task
- Supervised fine-tuning
  - Small corpus
  - Possibly small additional layer
  - 11 tasks in the original paper

BERT: pre-training

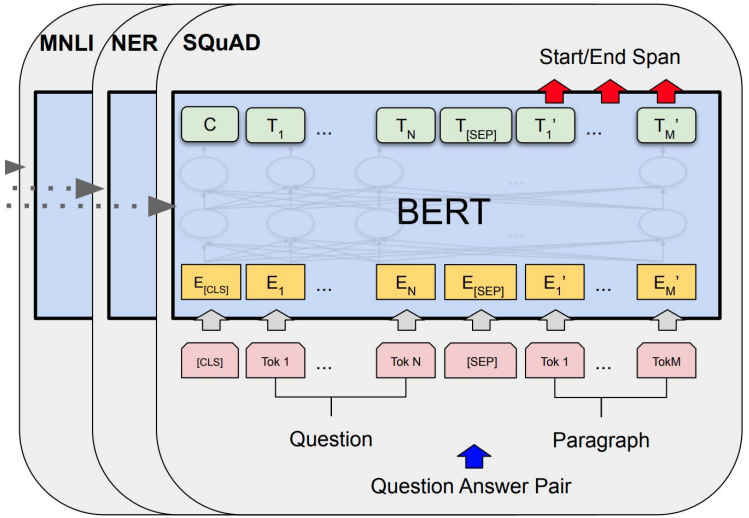
- Arbitrary tasks
- Begin sequence with [CLS]
- Masked LM
  - Special token [MASK]
  - Replace some words with [MASK]
  - Predict token from output representation (linear + softmax)
- Next sentence prediction
  - Separate sentences with [SEP]
  - Are the 2 sentences following each other?
  - From the [CLS] embedding



BERT fine-tuning (from the original paper)

- Tested on 11 different tasks
  - 1 or 2 sentences on input (with [SEP])
  - Per-token prediction or per-sentence (with [CLS])
  - + linear layer + softmax

- Example task: SQuAD
  - Find answer to question in a sentence / paragraph
  - Classifier for first token
  - Classifier for last token



[SQuAD: 100,000+ Questions for Machine Comprehension of Text, Rajpurkar, ArXiV'16]

BERT examples

Pre-training

Input = `[CLS] the man went to [MASK] store [SEP]`
`he bought a gallon [MASK] milk [SEP]`
Label = `IsNext`

Input = `[CLS] the man [MASK] to the store [SEP]`
`penguin [MASK] are flight ##less birds [SEP]`
Label = `NotNext`

Fine-tuning for SQuAD

In meteorology, precipitation is any product of the condensation of atmospheric water vapor that falls under **gravity**. The main forms of precipitation include drizzle, rain, sleet, snow, **graupel** and hail... Precipitation forms as smaller droplets coalesce via collision with other rain drops or ice crystals **within a cloud**. Short, intense periods of rain in scattered locations are called "showers".

What causes precipitation to fall?
gravity

What is another main form of precipitation besides drizzle, rain, snow, sleet and hail?
graupel

Where do water droplets collide with ice crystals to form precipitation?
within a cloud

Sentence embeddings for retrieval

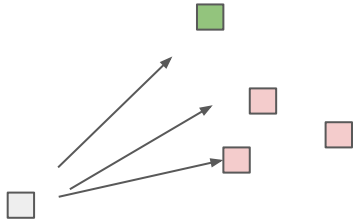
Contrastive learning

- f_\theta is a BERT
  - Average pooling of output token representations
- InfoNCE loss

$$s(q, d) = \langle f_\theta(q), f_\theta(d) \rangle.$$

$$\mathcal{L}(q, k_+) = -\frac{\exp(s(q, k_+)/\tau)}{\exp(s(q, k_+)/\tau) + \sum_{i=1}^{K} \exp(s(q, k_i)/\tau)},$$

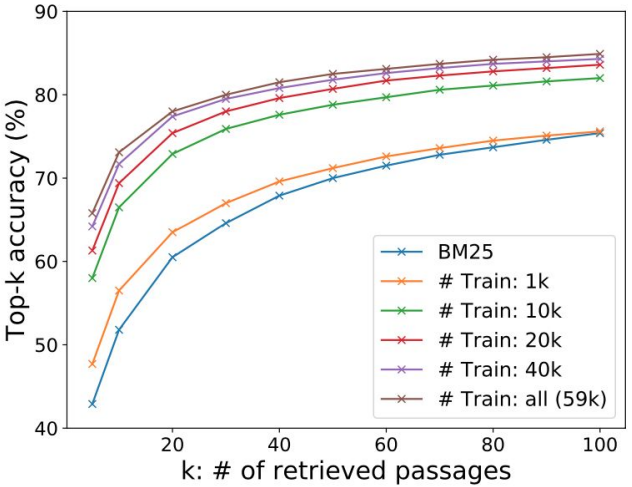
- Computed per training batch
- Form of contrastive learning
  - Expressed as a classification objective
  - Degrees of freedom: positives ? negatives?



[Dense passage retrieval for Open-Domain Question answering, Karphukhin etal, ArXiv‘20]

DPR – supervised

- Positive pairs:
  - (Question, answer) pair
- Negative pairs:
  - pairs from the same batch with high BM25 overlap



[Unsupervised dense information retrieval with contrastive learning, Izacard et al, ICML'22]

Contriever – unsupervised

- Positive pairs: independent cropping
  - Sometimes overlaps

> It is a truth universally acknowledged, that a single man in possession of a good fortune must be in want of a wife. However little known the feelings or views of such a man may be on his first entering a neighbourhood, this truth is so well fixed in the minds of the surrounding families, that he is considered as the rightful property of some one or other of their daughters.

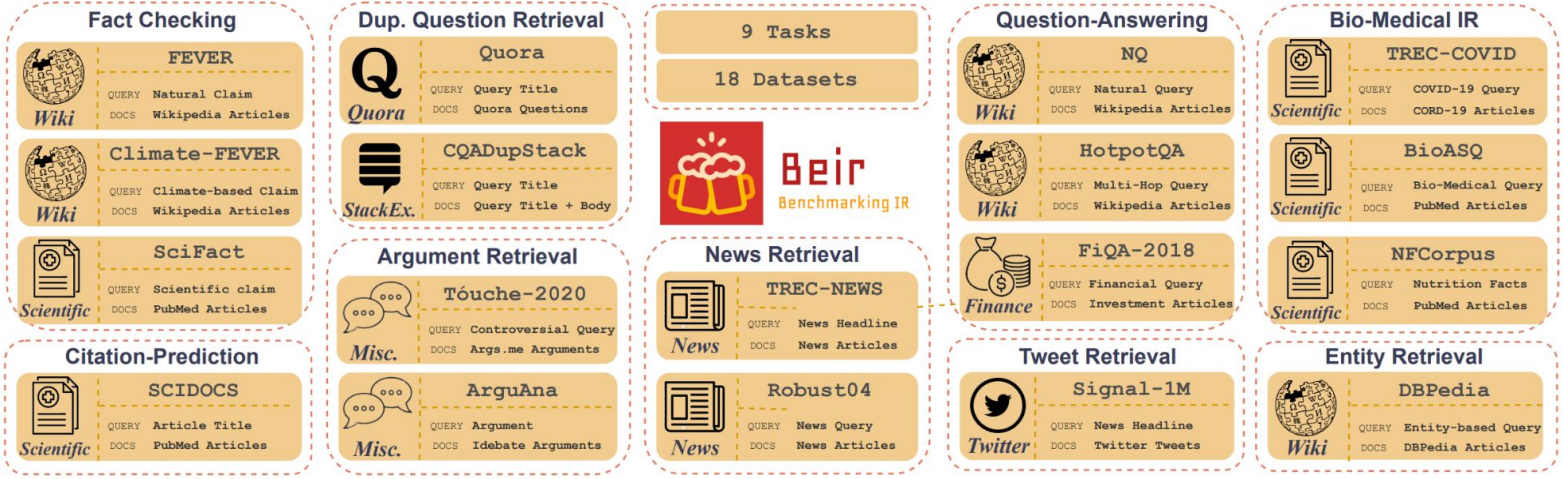
- Negative pairs: momentum contrast
  - previous batches
  - Embeddings computed with slowly updating network

$$\theta_k \leftarrow m\theta_k + (1-m)\theta_q$$

[Beir: A heterogenous benchmark for zero-shot evaluation of information retrieval models, Thakur et al, ArXiV'21]

Contriever results

BEIR family of benchmarks (doc retrieval)



FEVER [60] The Fact Extraction and VERification dataset is collected to facilitate the automatic fact checking. We utilize the original paper splits as queries **Q** and retrieve evidences from the pre-processed Wikipedia Abstracts (June 2017 dump) as our corpus **T**.

Climate-FEVER [14] is a dataset for verification of real-world climate claims. We include the original dataset claims as queries **Q** and retrieve evidences from the same FEVER Wiki corpus **T**. We manually included few Wikipedia articles (25) missing from our corpus, but present within our relevance judgements.

SciFact [68] verifies scientific claims using evidence from the research literature containing scientific paper abstracts. We use the original publicly available dev split from the task containing 300 queries as our test queries **Q**, and include all documents from the original dataset as our corpus **T**.

Contriever results

- Also multilingual
  - Common embeddings
- DRAGON
  - Extensive study of data augmentations

[How to Train Your DRAGON: Diverse Augmentation Towards Generalizable Dense Retrieval, Lin et al. ArXiV'23]

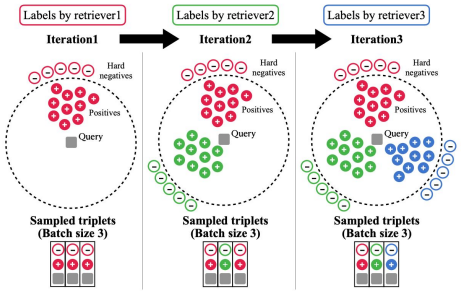


Figure 2: Illustration of progressive label augmentation. For each iteration of training, additional relevance labels from a teacher are augmented in the training data. By contrast, uniform supervision directly exposes models to all the supervisions (as in iteration 3) in the beginning.

Table 2: **BEIR Benchmark.** We report nDCG@10 on the test sets from the BEIR benchmark for bi-encoder methods without re-ranker. We also report the average and number of datasets where a method is the best ("Best on") over the entire BEIR benchmark (excluding three datasets because of their licence). Bold is the best overall. MS MARCO is excluded from the average. "CE" refers to cross-encoder.

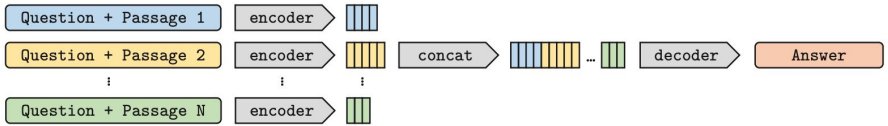
	BM25	BM25+CE	DPR	ANCE	TAS-B	Gen-Q	ColBERT	Splade v2	Ours	Ours+CE
MS MARCO	22.8	41.3	17.7	38.8	40.8	40.8	40.1	43.3	40.7	**47.0**
Trec-COVID	65.6	**75.7**	33.2	65.4	48.1	61.9	67.7	71.0	59.6	70.1
NFCorpus	32.5	**35.0**	18.9	23.7	31.9	31.9	30.5	33.4	32.8	34.4
NQ	32.9	53.3	47.4	44.6	46.3	35.8	52.4	52.1	49.8	**57.7**
HotpotQA	60.3	70.7	39.1	45.6	58.4	53.4	59.3	68.4	63.8	**71.5**
FiQA	23.6	34.7	11.2	29.5	30.0	30.8	31.7	33.6	32.9	**36.7**
ArguAna	31.5	31.1	17.5	41.5	42.9	**49.3**	23.3	47.9	44.6	41.3
Touche-2020	**36.7**	27.1	13.1	24.0	16.2	18.2	20.2	36.4	23.0	29.8
CQADupStack	29.9	37.0.	15.3	29.6	31.4	34.7	35.0	-	34.5	**37.7**
Quora	78.9	82.5	24.8	85.2	83.5	83.0	85.4	83.8	**86.5**	82.4
DBPedia	31.3	40.9	26.3	28.1	38.4	32.8	39.2	43.5	41.3	**47.1**
Scidocs	15.8	16.6	7.7	12.2	14.9	14.3	14.5	15.8	16.5	**17.1**
FEVER	75.3	**81.9**	56.2	66.9	70.0	66.9	77.1	78.6	75.8	**81.9**
Climate-FEVER	21.3	25.3	14.8	19.8	22.8	17.5	18.4	23.5	23.7	**25.8**
Scifact	66.5	68.8	31.8	50.7	64.3	64.4	67.1	**69.3**	67.7	69.2
Avg. w/o CQA	44.0	49.5	26.3	41.3	43.7	43.1	45.1	50.6	47.5	51.2
Avg.	43.0	48.6	25.5	40.5	42.8	42.5	44.4	-	46.6	50.2
Best on	1	3	0	0	0	1	0	1	1	9

Generating answers

The task

[Leveraging Passage Retrieval with Generative Models for Open Domain Question Answering Izacard & Grave, ArXiV'20]

- Q&A
- Make a real answer



- Insert special tokens
- Supervised training of the seq2seq model
  - Pretrained T5 model (220-770M params)
  - Train on 64 GPUs

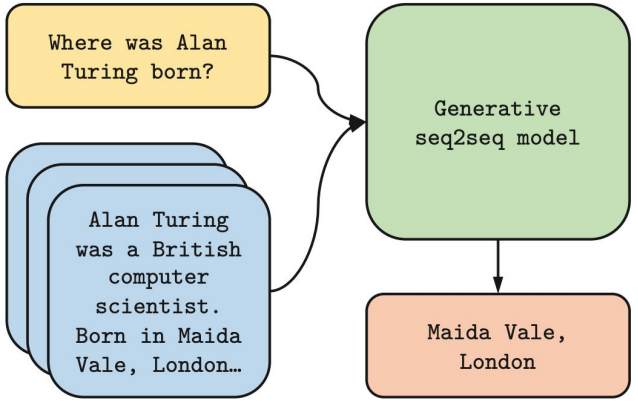
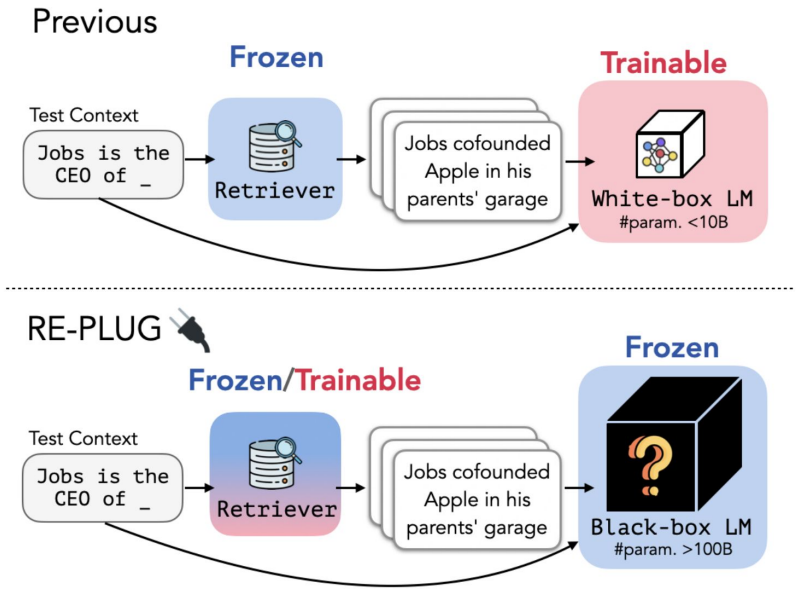


Figure 1: A simple approach to open domain question answering. First, it retrieves support text passages from an external source of knowledge such as Wikipedia. Then, a generative encoder-decoder model produces the answer, conditioned on the question and the retrieved passages. This approach scales well with the number of retrieved passages, as the performance keeps improving when retrieving up to one hundred passages.

RePlug

[Replug: Retrieval-augmented black-box language models, Shi et al, ArXiV'23]

TODO



Conclusion

- Embeddings in the text domain
- Word-level / sentence-level
- Tools
  - Metric learning
  - NCE
  - Pre-training / fine-tuning
  - Transformer models
- Natural way for information retrieval in a corpus of text docs

End