Dense retrieval using word embeddings

CS6101

Learning word and text representations

- Unsupervised (AD)
  - Word2vec, GloVe (next)
- Contextualized
  - ConvNet, RNN, Transformer (later)
- Pretrained then fine-tuned

Whole doc → shorter windows

- Whole doc co-occurrence signal is quite noisy
- Instead slide a window through text
  - Center word is w
  - Context words in window are C
- If we hide w, can we predict it from C?
- Model for cooccurrence of w and $c \in C$
  - Word2vec (Google)
  - GloVe (Stanford)

word2vec

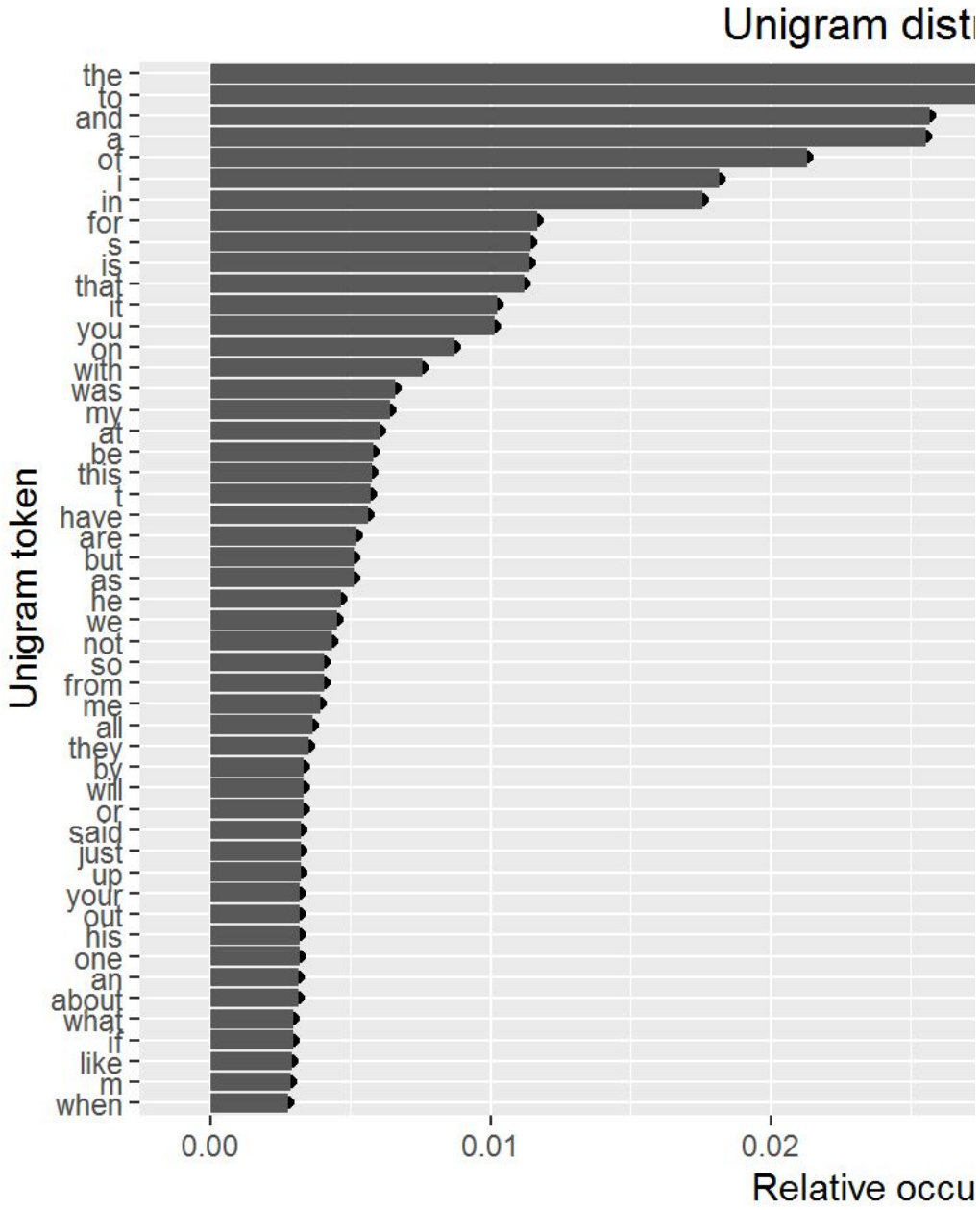
- Doc shortened to sliding window 'passage' centered on focus word f
- Cooccurring with other context words c
- For each word w, fit two vectors $\boldsymbol{u}_w, \boldsymbol{v}_w$
  - $\boldsymbol{u}_w$ for focus 'role', $\boldsymbol{v}_w$ for context 'role'
- Model probability that
  - f and c cooccur as $\Pr(f, c) = \sigma(\boldsymbol{u}_f \cdot \boldsymbol{v}_c)$
  - f and a negative sample word $\overline{c}$ do not cooccur as $\Pr(f, \neg\overline{c}) = 1 - \sigma(\boldsymbol{u}_f \cdot \boldsymbol{v}_{\overline{c}}) = \sigma(-\boldsymbol{u}_f \cdot \boldsymbol{v}_{\overline{c}})$
- Fit U, V to maximize $\prod_f \prod_c \sigma(\boldsymbol{u}_f \cdot \boldsymbol{v}_c) \prod_{\overline{c}} \sigma(-\boldsymbol{u}_f \cdot \boldsymbol{v}_{\overline{c}})$
- Nonconvex, sensitive to how (many) $\overline{c}$ are sampled per f

$\frac{1}{\quad}$ is the

There are a few variations on the form of the model, but they show comparable behavior

Negative sampling for word2vec

- For each $\langle f, c \rangle$, collect K samples $\langle f, \overline{c} \rangle$ where $\overline{c}$ does not occur in current context
- Note, negative $\overline{c}$ in one context can be positive c in another
- Natural to sample unigram distrib q_w
- Any f, f' similar to the, to, and, a, etc. therefore $f \approx f$
- Helps to flatten unigram distribution to $p(w) \propto q_w^{0.75}$ (plus some magic)
- No theory to justify the new gravitational constant!



Batching for word2vec

- Repeated window scans through corpus
  - (Not really necessary, as shown in GloVe)
- A batch is collected from a focus word f and several context group, each having
  - One postive context word c
  - Several (K) negative context words $\overline{c}$
- Notation $\langle f, \{\langle c, \{\overline{c}\}\rangle\}\rangle$; $\{\overline{c}\} = \{\overline{c}_1, \cdots, \overline{c}_K\}$
- Find gradient of objective wrt these words, and update with small step size
- Flattening to $\langle f, c, \overline{c}\rangle$ and shuffling destroys optimization!

GloVe

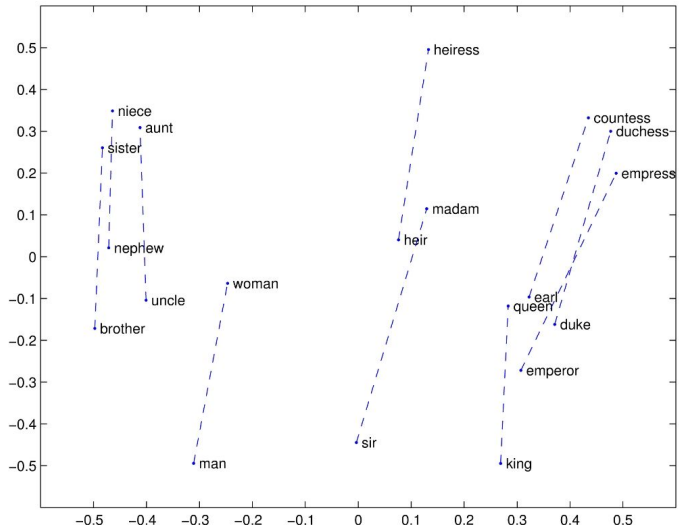
- Collect global word cooccurrence (in window) counts $n(w, w')$
  - Collect top counts ("heavy hitters") in $o(W^2)$ space, one/few scans over corpus
- Parameter space with four components
  - Compared to word2vec's two
  - U, V as in word2vec (direction of word)
  - Plus biases $a_w,\ b_w \in \mathbb{R}$ for each word (marginal popularity)
- Model $\log n(w, w') = \boldsymbol{u}_w \cdot \boldsymbol{v}_{w'} + a_w + b_{w'}$, with loss

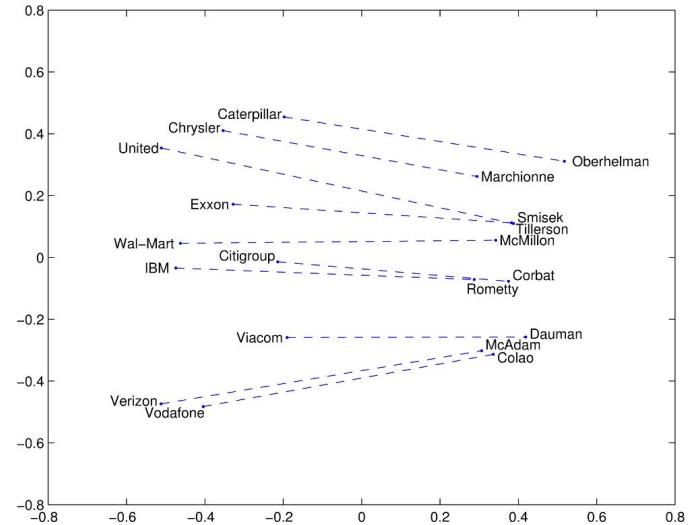
$$\sum_{w,w'} \text{squash}(n(w,w'))[\boldsymbol{u}_w \cdot \boldsymbol{v}_{w'} + a_w + b_{w'} - \log n(w,w')]^2$$

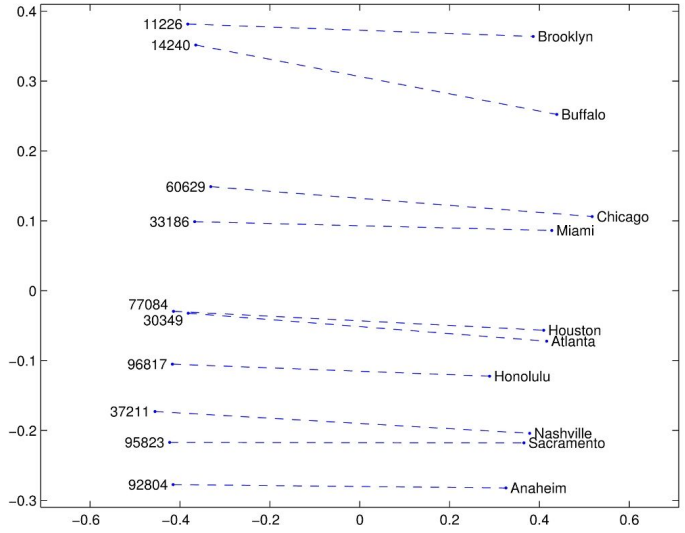
- Parameters fitted by gradient descent
- But no corpus scan or negative sampling during optimization

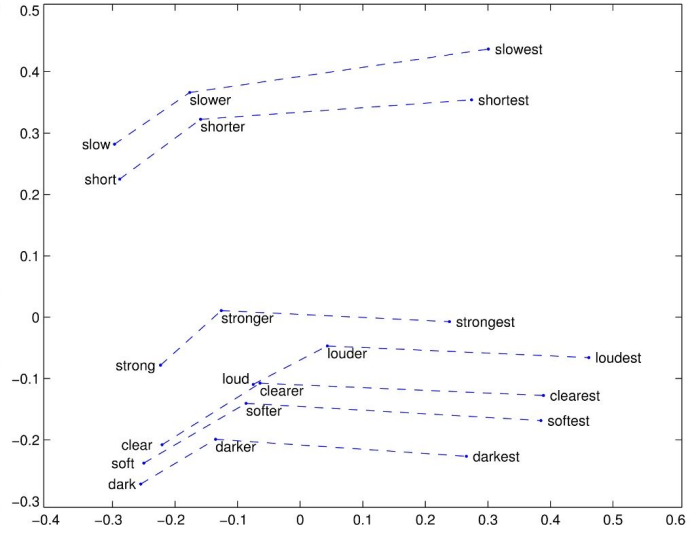
Surprising property of word vectors

- Word embeddings were known since LSI
- Novelty of word2vec and GloVe partly in pointing out that abstract relations are represented by translations in space
- (Surprising, given objective involved only dot products of unnormalized vectors)
- Analogy games
  - India : Delhi :: Australia : ?
- Also played a role in early knowledge graph embeddings







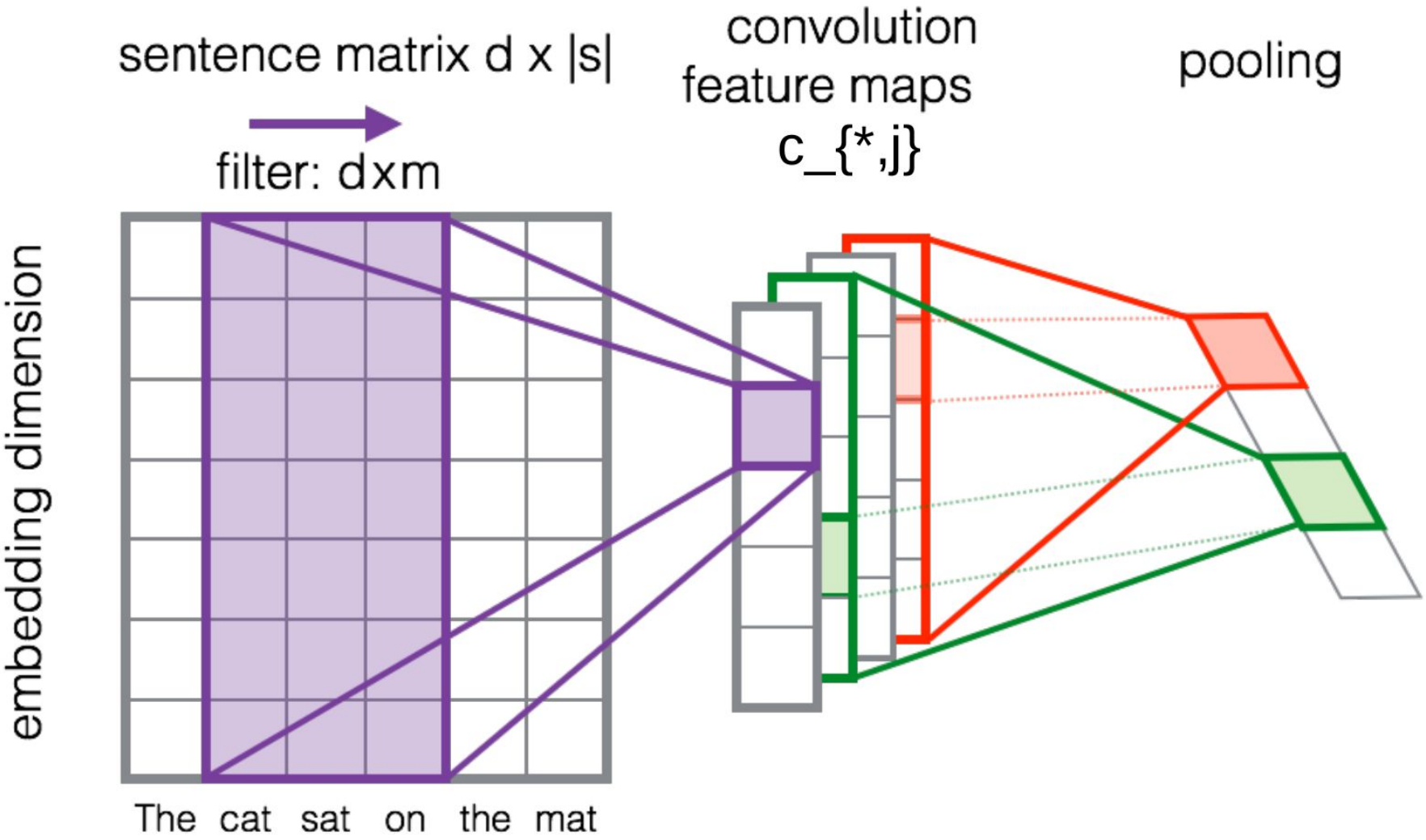


Sentence/passage/doc representation

- LSI and PSI simultaneously computed embeddings for each word and doc
  - New docs have to be “folded in” via EM
  - Cart before horse vs. horse before cart
  - Word embeddings are context independent
- Easiest: weighted average of word vectors
- Sliding window and convolution
- Memory (RNN) and attention (transformers)

1d convnet for text

- Hierarchical position-aware aggregation without parsing



1d convnet, multiple channels

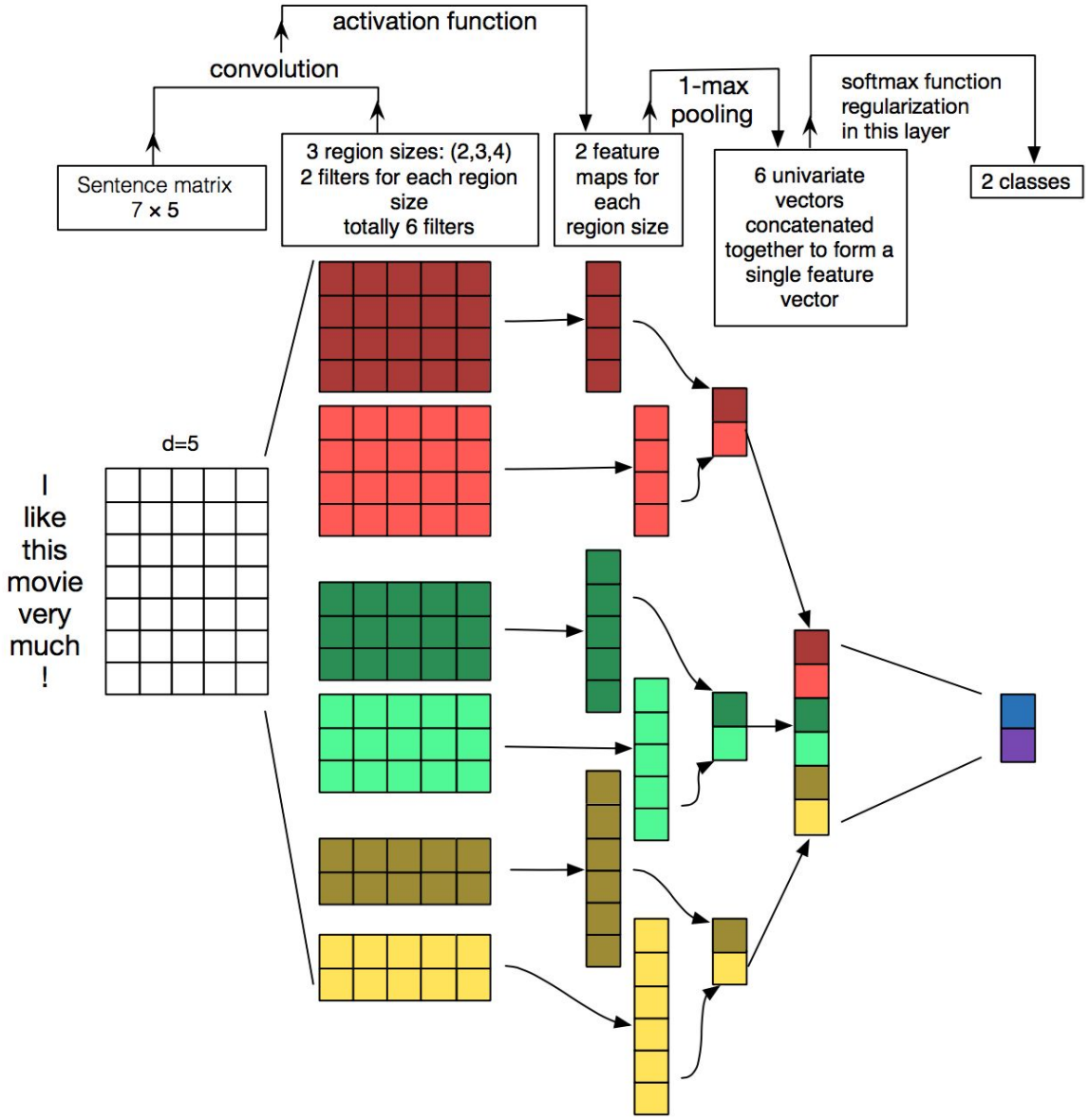
- $d \times |s|$ word embedding matrix S for sentence with $|s|$ words, here 7×6
- 7×3 'stencil' slides through four positions
- Weight matrix is $W \in \mathbb{R}^{7\times3}$
- When left edge of W is positioned at column k of S, we compute $c_k = \sum_{i=0}^{6} \sum_{j=0}^{2} W[i,j]\ S[i, j+k]$
- This gives c, a vector with 4 numbers
- In place of a single W, use 'channels' or 'heads' W, W, W, W, W
- Leading to c, c, c, c, c i.e., matrix $C \in \mathbb{R}^{4\times5}$
- Then another 'layer' (vector) $u \in \mathbb{R}^5$, leading to output $f = Cu \in \mathbb{R}^4$

Expectation: different channels can be trained to pick up different kinds of signals in text

Applications of passage representation

- (Not just for convnet)
- Learn a classifier whose input is f, to predict labels like topic, sentiment, product rating, spam/not-spam, etc.
- If there are M labels, use matrix $V \in \mathbb{R}^{4\times M}$; output $\mathrm{softmax}(fV)$
- Compare two passages s_1, s_2 for 'semantic' similarity $s_1 \cong s_2$ or entailment $s_1 \Longrightarrow s_2$
  - Common use in search
- If unsure of best shape of sliding stencil, use several stencils of various sizes

1d convnet, multiple stencils



Contextual word embeddings

He swam near the bank because the current was swift.

‘Encoder’

His bank offered a low interest on current accounts.

‘Encoder’

Collective ‘meaning’ disambiguation

Two paradigms:

- Left to right influence (RNN, LSTM)
- All to all influence (attention)

HMM, MEMM

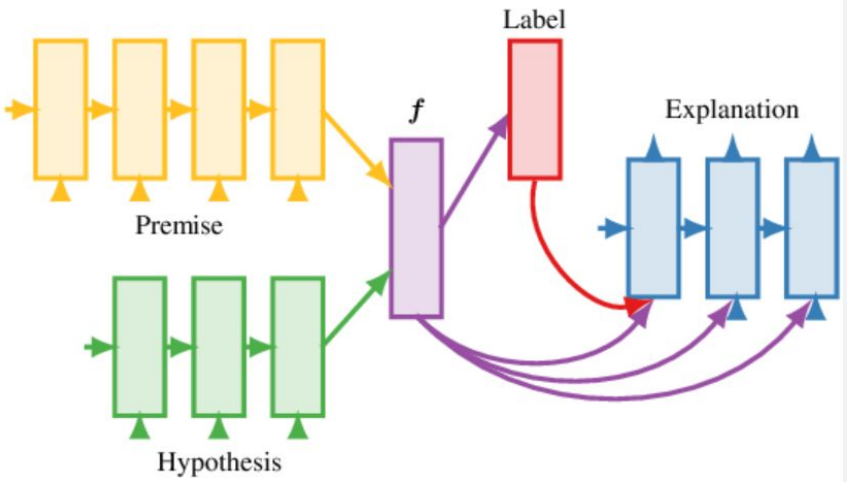
- Observed input symbols $x_{1:T}$, $x_t \in [F]$
- Inferred output 'states' $y_{0:T}$ (sentinel y_0); $y_t \in [M]$
- Classic applications: POS, NE tagging
- Hidden Markov model
  - $\Pr(x_{1:T}, y_{1:T}) = \prod_{t=1}^{T} \Pr(y_{t-1} \to y_t) \Pr(y_t \uparrow x_t)$
  - State transition '$\to$' and emission '$\uparrow$'
- Maximum entropy Markov model
  - $\Pr(y_{1:T} | x_{1:T}) = \prod_{t=1}^{T} \Pr(y_{t-1} \to y_t | x_t)$
  - Each term modeled as logistic regression
- RNNs are continuous state counterparts to MEMMs

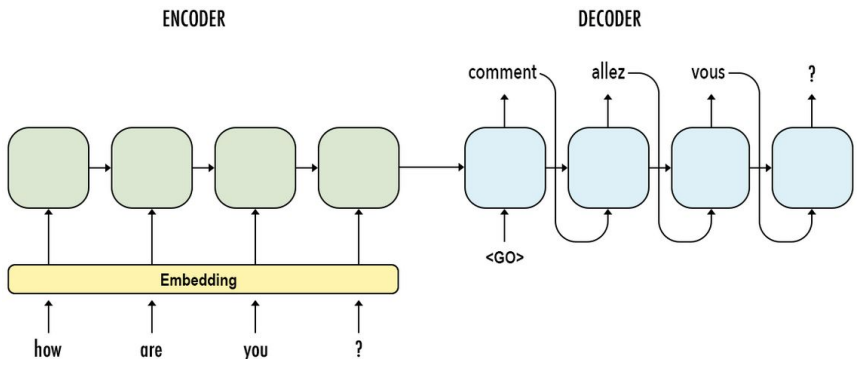
RNNs (GRU, LSTM) and bidirectional RNNs

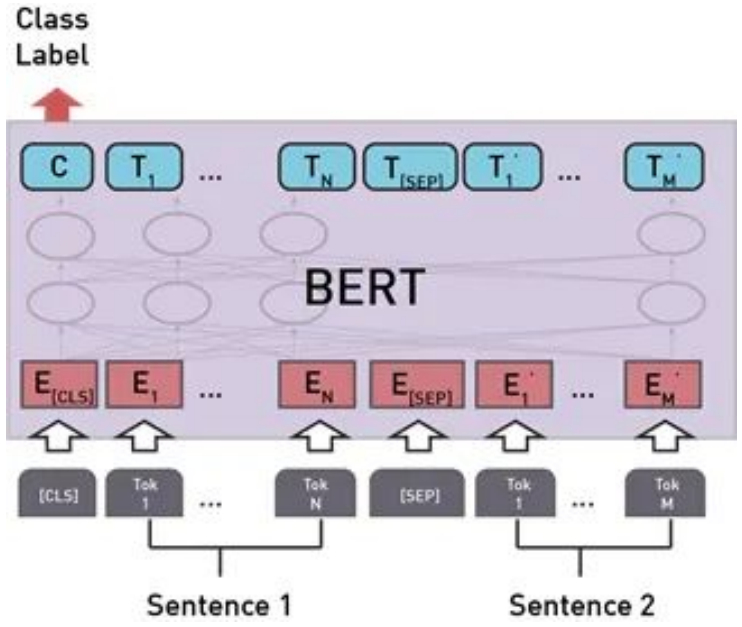
- (Brief review of cell logic)
- In left-to-right RNN, $\overrightarrow{h}_t$ is a compressed representation of $x_{1:t}$
- Similarly $\overleftarrow{h}_t$ for right-to-left RNN
- Together, $\overleftrightarrow{h}_t = \left[\overrightarrow{h}_t, \overleftarrow{h}_t\right]$ give a compressed representation of the whole text tailored to position t
  - Can choose to include or exclude info about x_t itself
- Various tasks can be solved using $\overleftrightarrow{h}_{1:T}$
  - Pooled $\bigoplus_t \overleftrightarrow{h}_t$ can be used for classification; extend to sentence pair labeling
  - Can mask x_t and try to predict it using $\overleftrightarrow{h}_t$
  - Can give up on right-to-left signals and generate new text

Sentence pair tasks

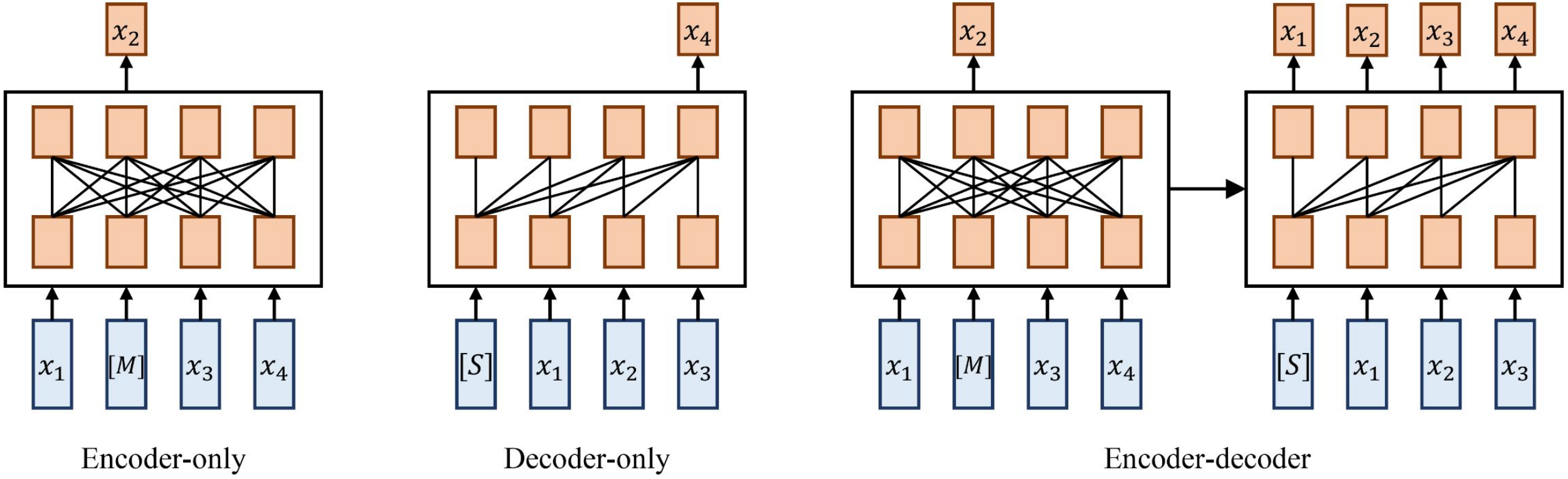
- Pair classification (e.g. implication, next sentence)
- Translation (primed generation)
- Question answering (can be regarded as either classification or generation)







Sequence network nomenclature

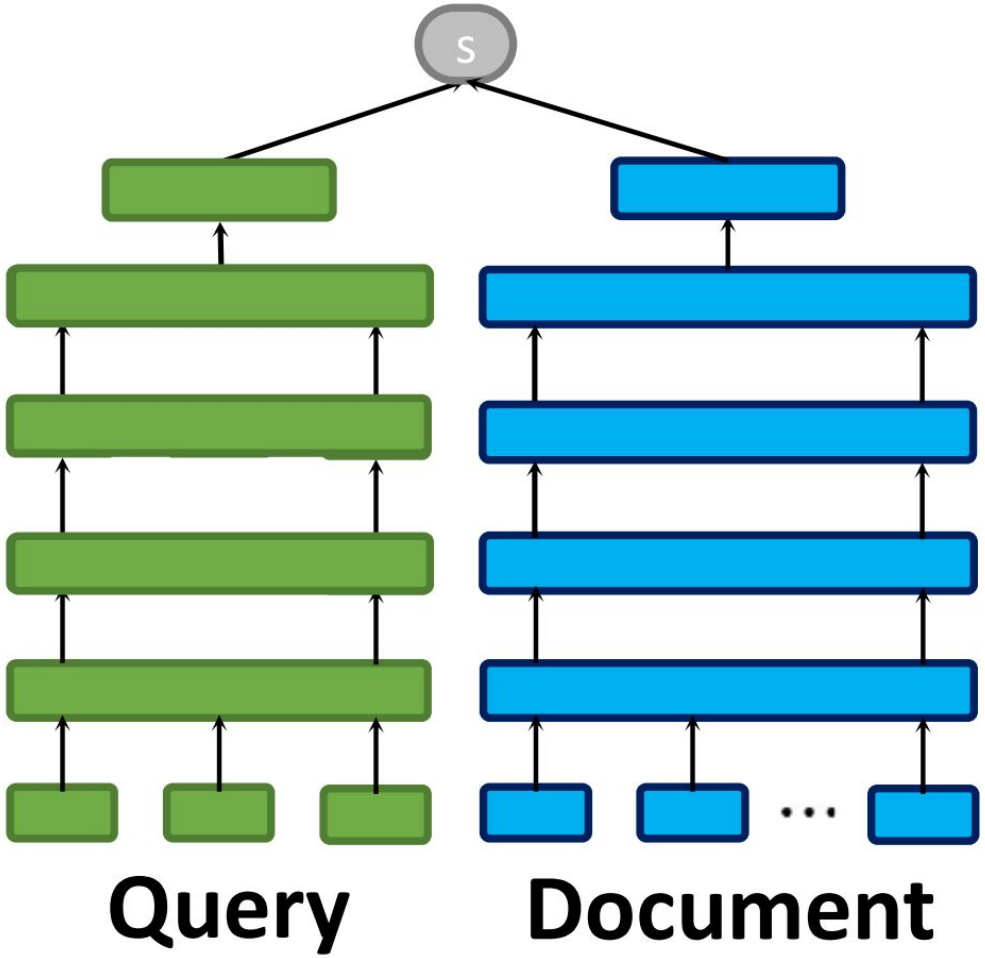


- Classification/tagging, generation, translation
- Encoder can look ahead, decoder cannot

Next…

- Late interaction single vector retrieval
  - LSH for various distance/similarity scores
- Other ANN techniques (HNSW)
- Trainable (dense) retriever
  - aka “dense passage retrieval” or DPR
- Late interaction multi-vector retrieval
  - Middle ground between single-vector and early interaction
- Generative rerankers
- Generative retrievers (DSI)

Deep retrieval: late interaction



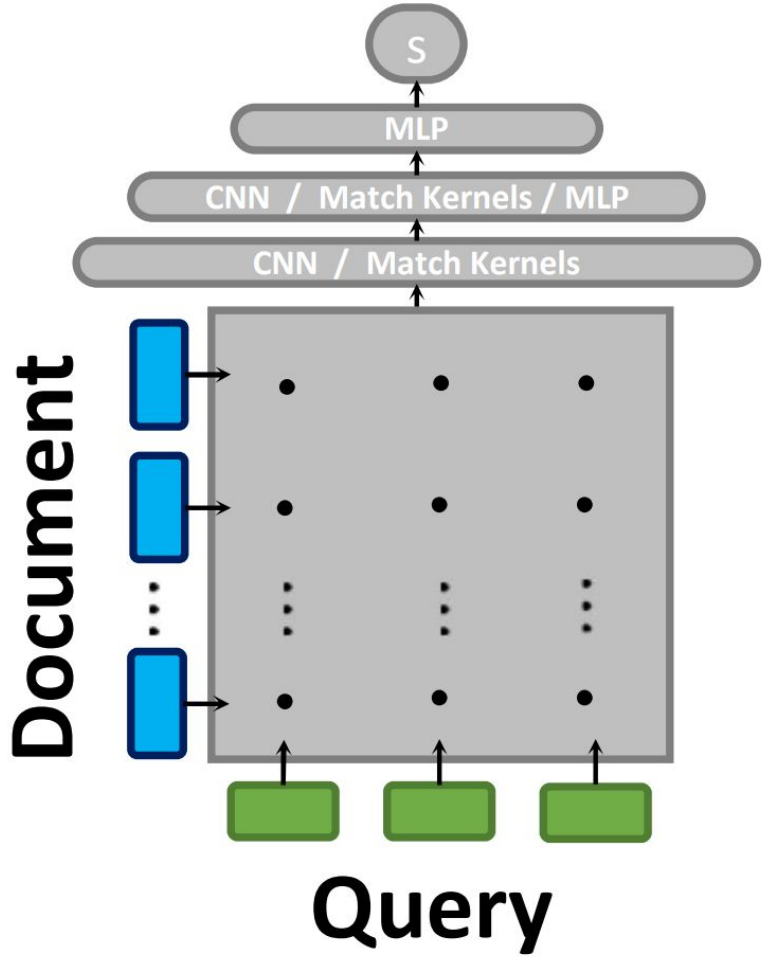
(a) Representation-based Similarity

(e.g., DSSM, SNRM)

− Too much compression

+ Fast similarity search

Deep retrieval: crossbar

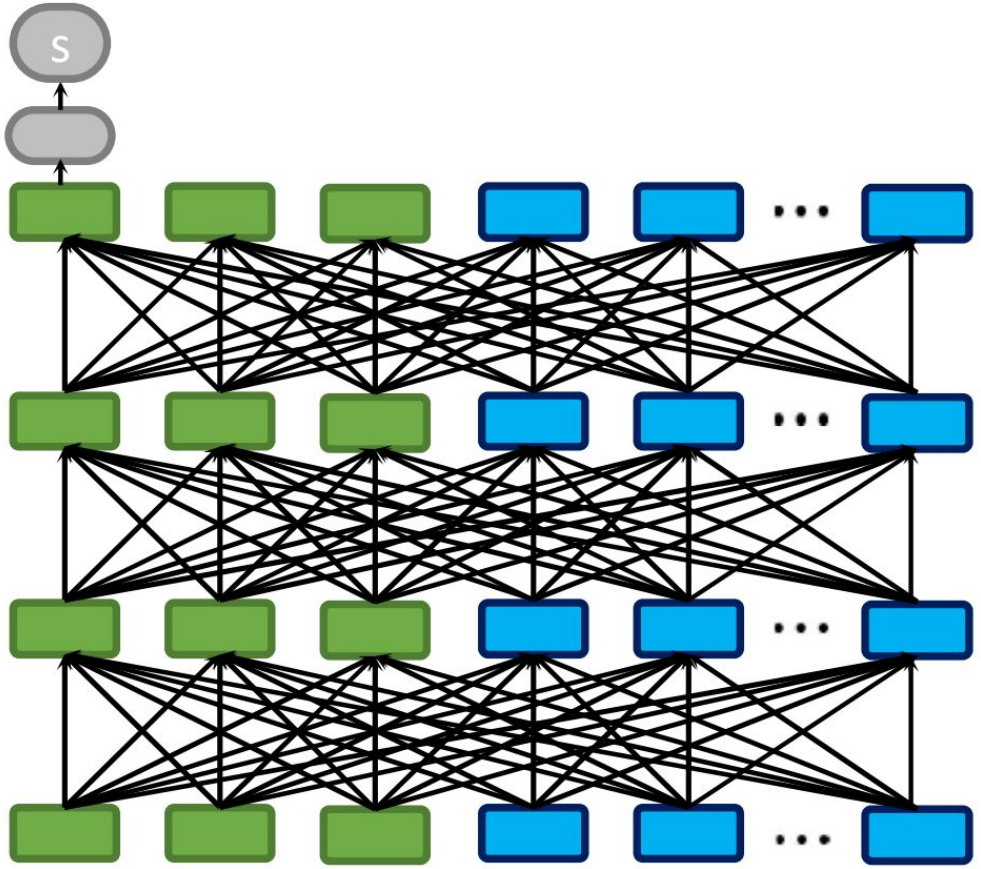


(b) Query-Document Interaction

(e.g., DRMM, KNRM, Conv-KNRM)

- All (promising) docs to be checked; slow
- No self-attention

All-to-all early interaction

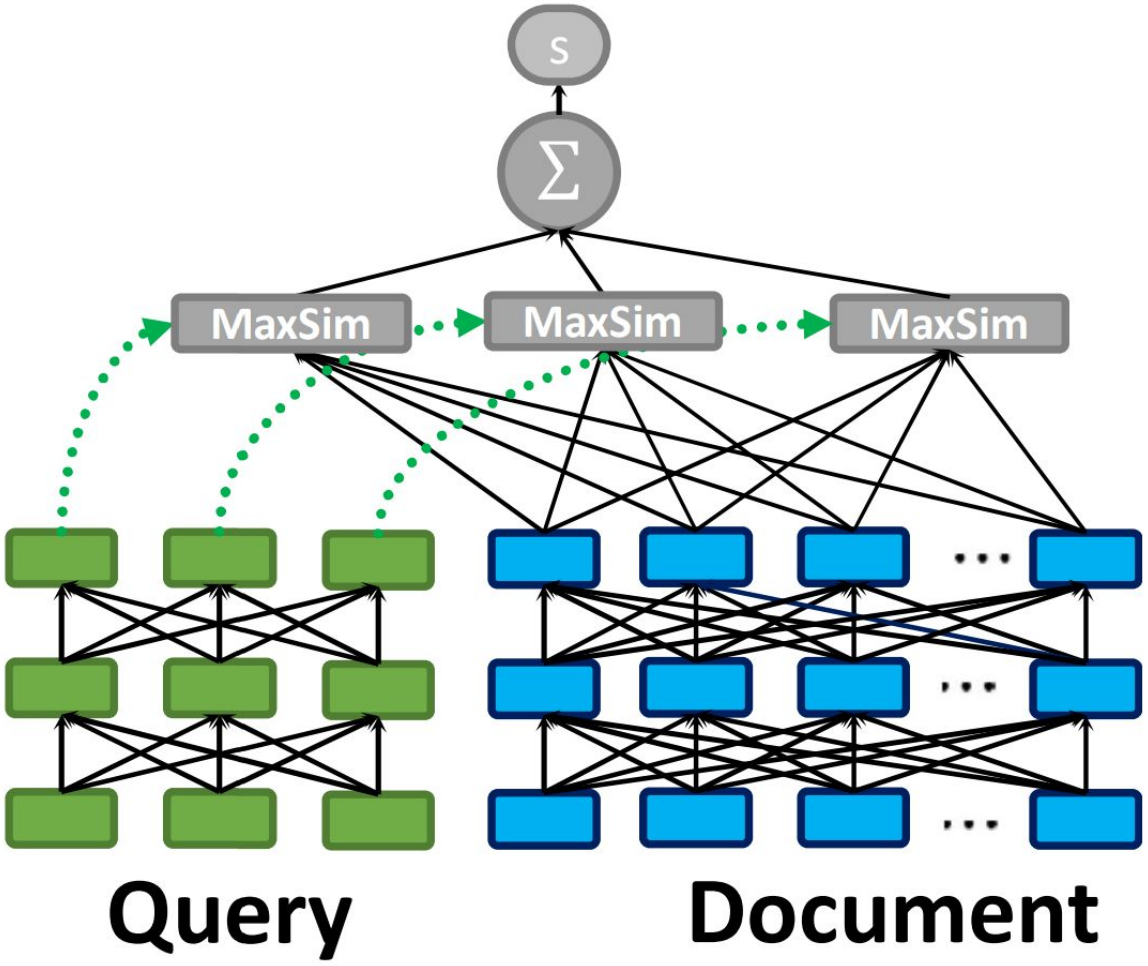


(c) All-to-all Interaction

(e.g., BERT)

- Even slower
- Soft vs hard matches?
- Position embeddings, layout tricks

Late, sparse interaction (ColBERT)



(d) Late Interaction

(i.e., the proposed ColBERT)

Separate stacks for q, d

$$E_q := \texttt{Normalize}(\ \texttt{CNN}(\ \texttt{BERT}(\text{“}[Q]q_0q_1\ldots q_l\#\#\ldots\#\text{”})\ )\ )$$

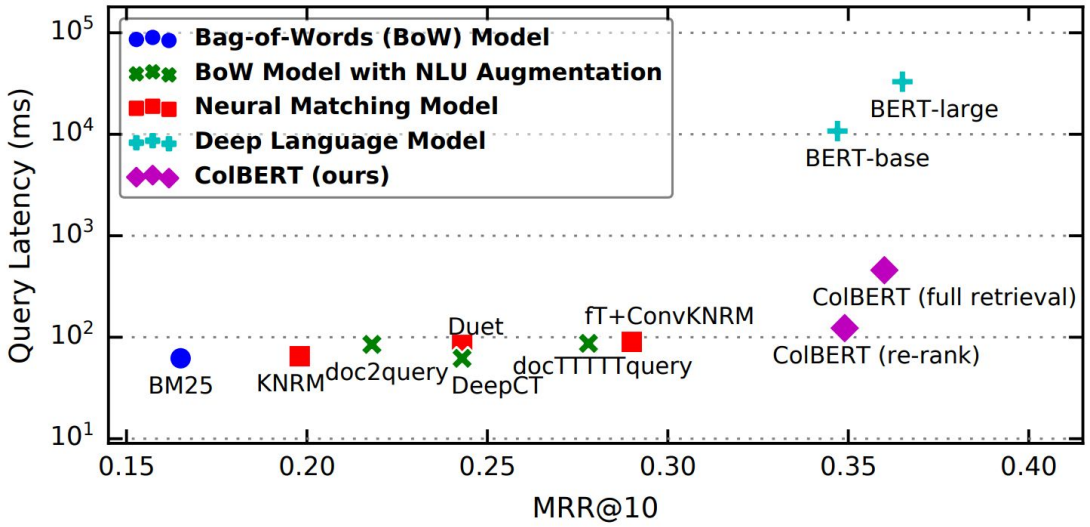
$$E_d := \texttt{Filter}(\ \texttt{Normalize}(\ \texttt{CNN}(\ \texttt{BERT}(\text{“}[D]d_0d_1\ldots d_n\text{”})\ )\ )\ )$$

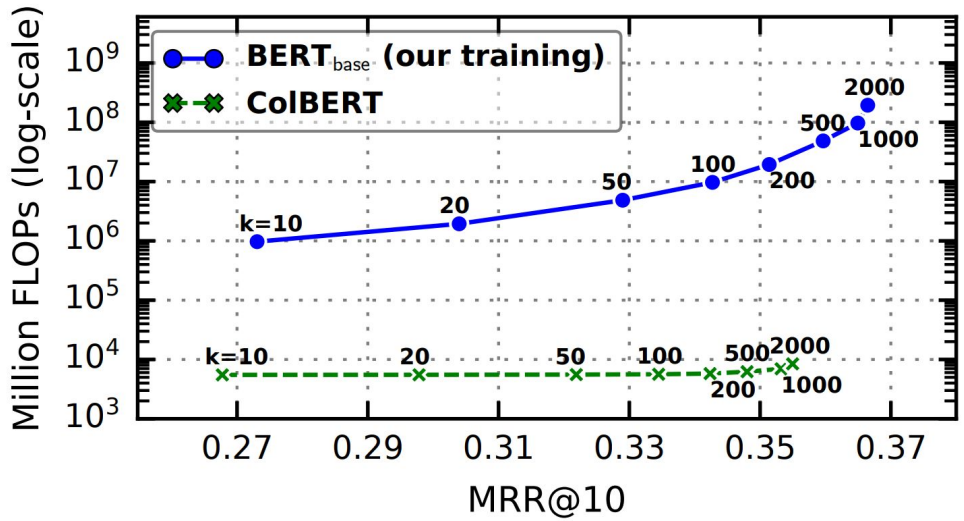
Scoring is not one-shot but sum over query tok

$$S_{q,d} := \sum_{i \in [|E_q|]} \max_{j \in [|E_d|]} E_{q_i} \cdot E_{d_j}^T$$

Each q token picks best d cover

Performance summary





Deep retrieval data set

- https://microsoft.github.io/msmarco/
- 100,000 real Bing questions
- Human generated answer
- Another 1,000,000 question dataset
- Tasks
  - Document ranking leaderboard
  - Passage retrieval leaderboard
  - Also QA and NL generation

References

- Huang+2013 bag-of-word-embeddings, symmetric aggregation, dot product similarity
- Shen+2014 use convnet instead of bag
- Palangi+2015 use LSTM instead of convnet
- Gillick+2018 avg word vectors; sampling and loss
- Nogueira+2019 use BERT instead of …
- Bi-encoder vs cross-encoder
- Karpukhin+2020 difficult negatives from BM25
- Xiong+2020 difficult negatives from model itself
- Humeau+2019 Khattab+2020 address cross-encoder inefficiency