



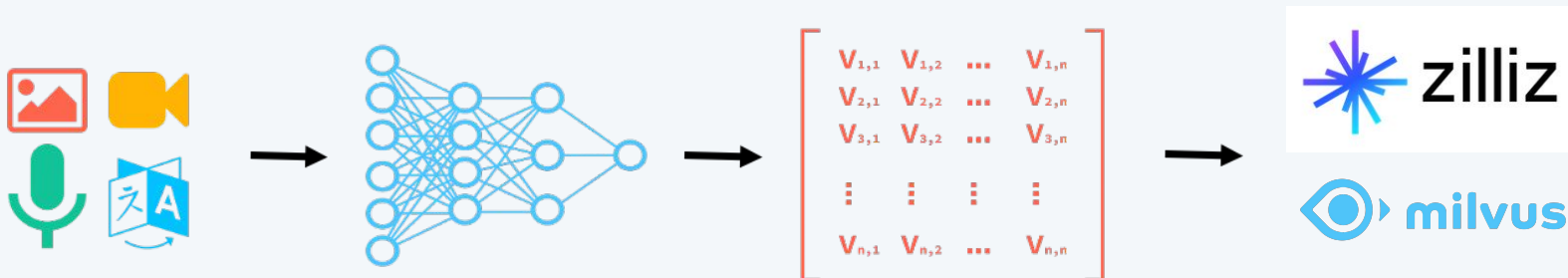
Sparse and Dense Embeddings

Frank Liu



A Quick Refresher

Vectors unlock unstructured data



**Knowledge Base
(Documents)**

Embedding Models

Vectors

Vector Databases

Embeddings models are workhorses of AI apps

Hugging Face Search models, datasets, users...

🔍 sentence-transformers **all-MiniLM-L6-v2** 📄 like 1.32k

🏷️ Sentence Similarity Sentence Transformers PyTorch TensorFlow Rust s2orc

📁 flax-sentence-embeddings/stackexchange_xml ms_marco gooaq yahoo_answers_topics

📁 code_search_net search_qa eli5 snli multi_nli wikihow natural_questions

📁 trivia_qa embedding-data/sentence-compression embedding-data/flickr30k-captions

📁 embedding-data/alltex embedding-data/simple-wiki embedding-data/QQP

📁 embedding-data/SPECTER embedding-data/PAQ_pairs embedding-data/WikiAnswers English

bert feature-extraction Inference Endpoints arxiv:1904.06472 arxiv:2102.07033

arxiv:2104.08727 arxiv:1704.05179 arxiv:1810.09305 License: apache-2.0

📄 Deploy Use in sentence-transformers


📄 Model card 📄 Files 🗨️ Community 42

📄 Edit model card

all-MiniLM-L6-v2

This is a [sentence-transformers](#) model: It maps

Downloads last month **5,523,635**



Hugging Face Search models, datasets, users...

🔍 meta-llama **Llama-2-7b-chat-hf** 📄 like 2.5k

🏷️ Text Generation Transformers PyTorch Safetensors English llama facebook

meta llama-2 text-generation-inference arxiv:2307.09288

📄 Train Deploy Use in Transformers

📄 Model card 📄 Files

🔗 Access Llama 2 on Hugging Face

This is a form to enable access to Llama 2 on Hugging Face after you have been granted access from Meta. Please visit the [Meta website](#) and accept our license terms and acceptable use policy before submitting this form. Requests will be processed in 1-2 days.

Your Hugging Face account email address MUST match the email you provide on the Meta website, or your request will not be approved.

Downloads last month **772,056**

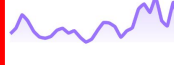
🔗 Safetensors

Model size 6.74B params

Tensor type FP16

🔗 Text Generation

Inference API has been turned off for this

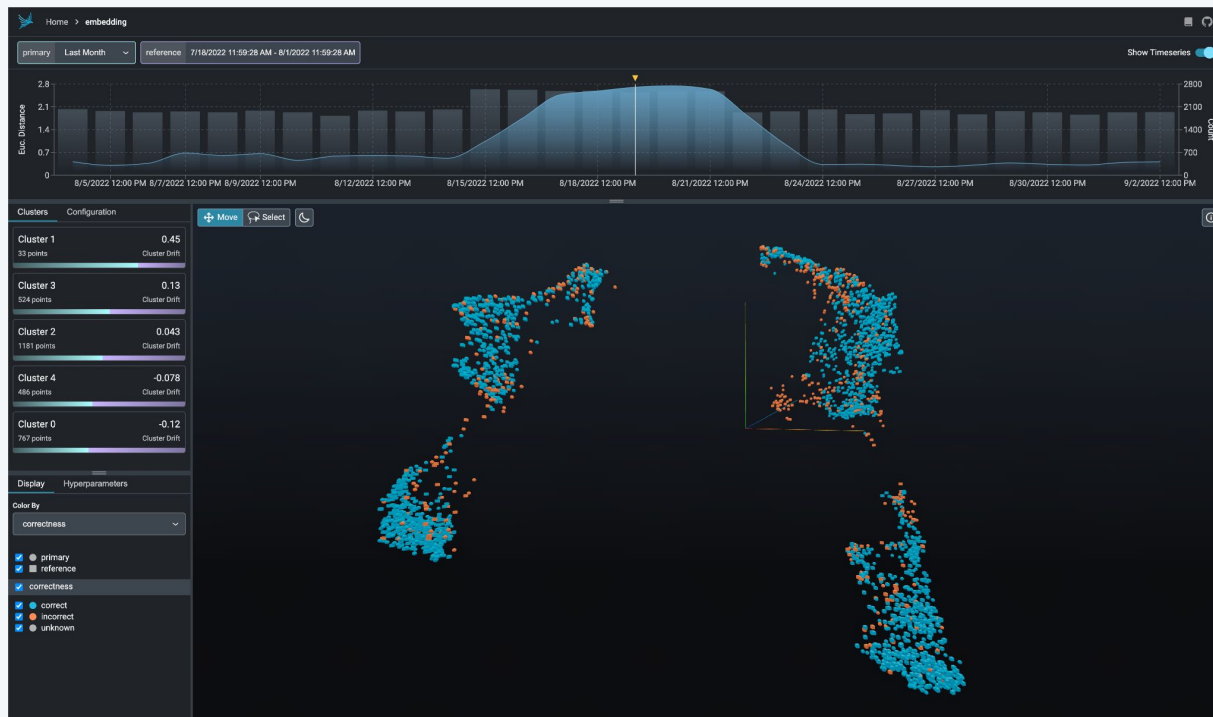


There are lots of embedding models out there

Rank ▲	Model ▲	Average ▲	AskUbuntuDupQuestions ▲	MindSmallReranking ▲	SciDocsRR ▲	StackOverflowDupQuestions ▲
1	e5-mistral-7b-instruct	60.21	66.98	32.6	86.33	54.91
2	ember-v1	60.04	64.46	32.27	87.56	55.85
3	bge-large-en-v1.5	60.03	64.47	32.06	87.63	55.95
4	UAE-Large-V1	59.88	64.2	32.51	87.49	55.32
5	sf_model_e5	59.86	64.32	32.27	87.47	55.4
6	voyage-lite-01-instruct	59.74	65.77	31.69	87.03	54.49
7	all-mpnet-base-v2	59.36	65.85	30.97	88.65	51.98
8	gte-large	59.13	63.06	32.63	87.2	53.63
9	bge-base-en-v1.5-quant	58.94	62.39	31.89	87.05	54.45
10	bge-base-en-v1.5-seqLen-384-bs-1	58.86	62.13	31.2	87.49	54.61
11	bge-base-en-v1.5	58.86	62.13	31.2	87.49	54.61
12	stella-base-en-v2	58.78	62.72	31.91	86.66	53.81

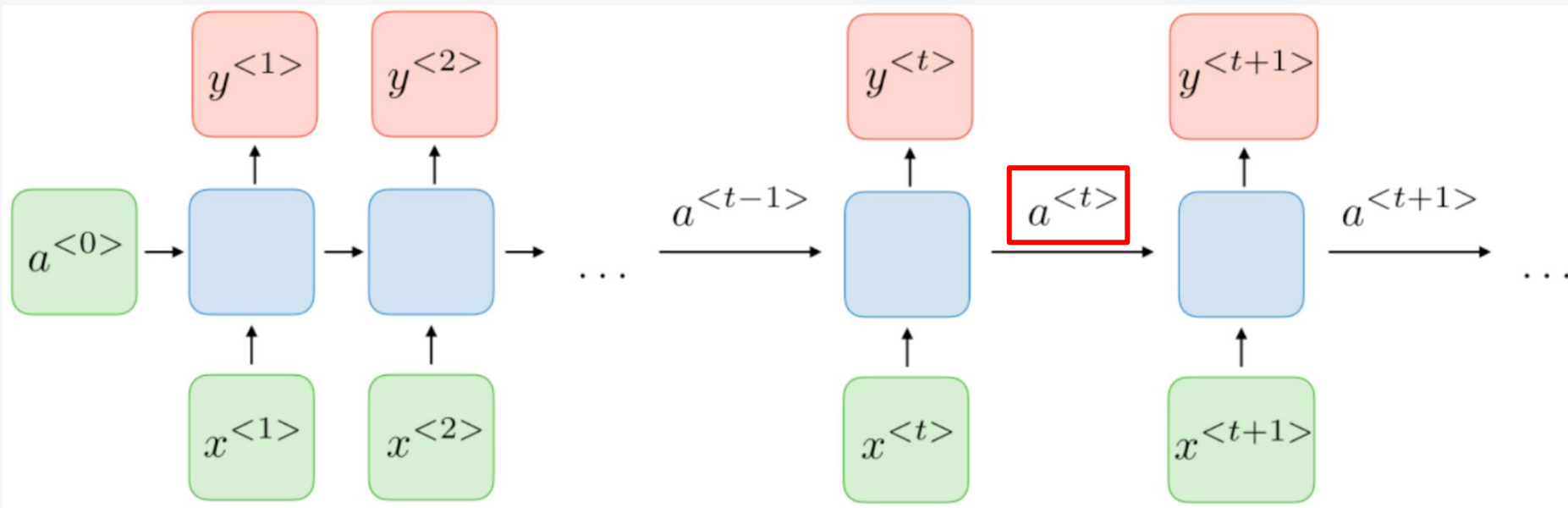
Source: [MTEB Leaderboard](#)

Visualizing dense embeddings



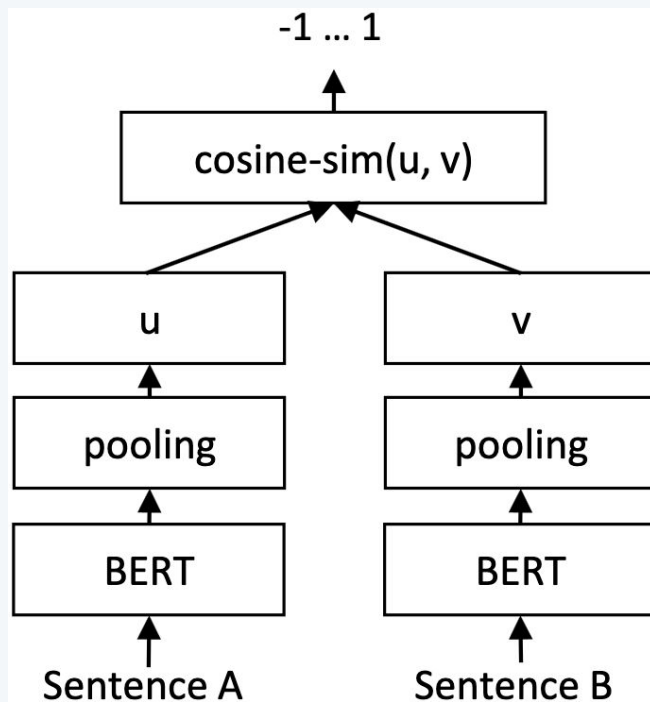
Source: [Arize Phoenix](#)

Recurrent neural networks



Source: [CS230 notes](#)

Sentence BERT



Source: [SBERT paper](#)

Bi-encoder
1 embedding per text

Sparse Embeddings

Dense embeddings are great...

Dense embeddings are great...

... but they lack lexical information

We're used to searching lexically, not semantically

We're used to searching lexically, not semantically

- Here are some common Google searches
 - “rubik’s cube algorithm”
 - “cups in a quart”
 - “how to tie a tie”

We're used to searching lexically, not semantically

- Here are some common Google searches
 - “rubik’s cube algorithm”
 - “cups in a quart”
 - “how to tie a tie”
- Keywords play an important role

We're used to searching lexically, not semantically

- **Here are some common Google searches**
 - “rubik’s cube algorithm”
 - “cups in a quart”
 - “how to tie a tie”
- **Keywords play an important role**
 - Lexical search is superior for out-of-domain data

We can combine sparse and dense embeddings

We can combine sparse and dense embeddings

Ranking	BM25 with title boosting	BM25 with content boosting	Semantic
1	Document-2	Document-3	Document-4
2	Document-3	Document-5	Document-2
3	Document-5	Document-2	Document-5
4	Document-1	Document-1	Document-3
5	Document-4	Document-4	Document-1

Let's calculate RRF for each document and rerank:

Document-1	$1/4 + 1/4 + 1/5 = 0.7$	→	Document-2
Document-2	$1/1 + 1/3 + 1/2 = 0.83$		Document-3
Document-3	$1/2 + 1/1 + 1/4 = 1.75$		Document-4
Document-4	$1/5 + 1/5 + 1/1 = 1.4$		Document-5
Document-5	$1/3 + 1/2 + 1/3 = 1.16$		Document-1

Source: [Sowmiya Jaganathan](#)

Sparse Embedding Algorithms

Pure lexical sparse embeddings (e.g. TF-IDF)

$$w_{x,y} = \text{tf}_{x,y} \times \log \left(\frac{N}{df_x} \right)$$

TF-IDF

Term x within document y

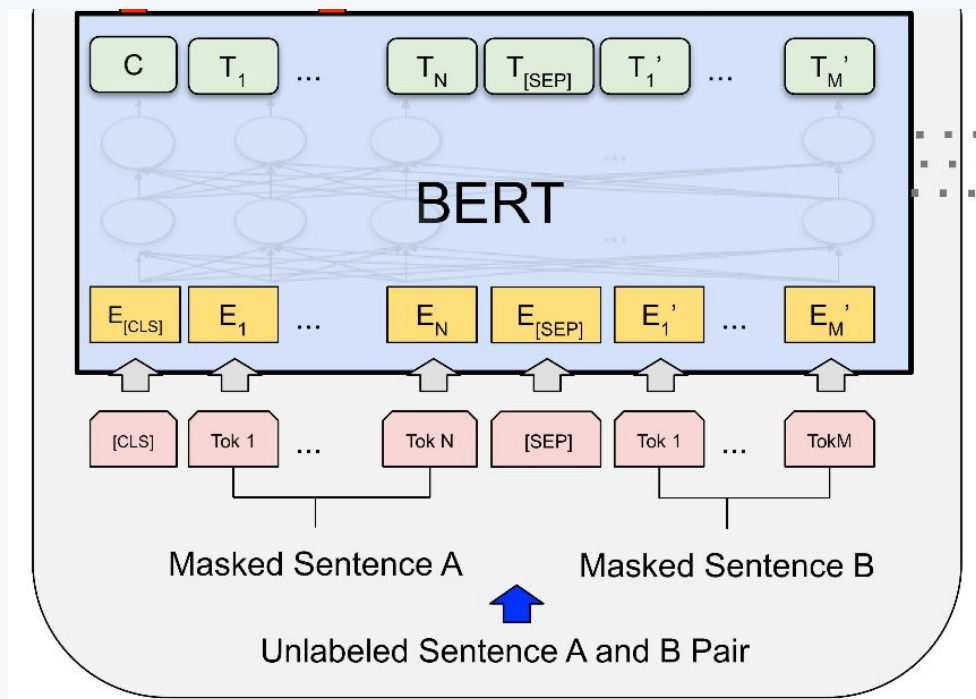
$\text{tf}_{x,y}$ = frequency of x in y

df_x = number of documents containing x

N = total number of documents

Source: [Ted Mei](#)

Bonus: CoBERT



Source: [BERT paper](#)

Bonus: ColBERT

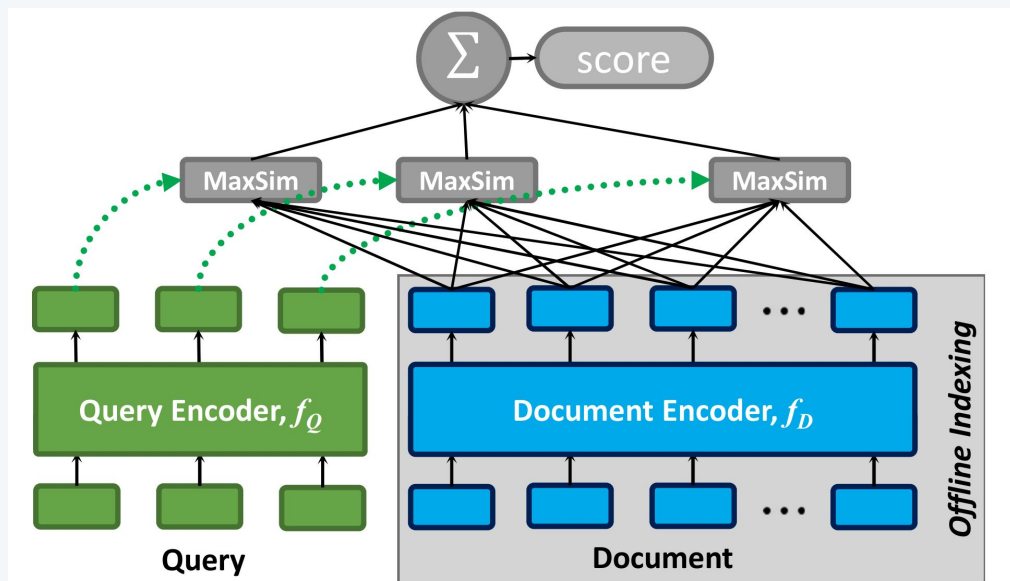


Figure 3: The general architecture of ColBERT given a query q and a document d .

Source: [ColBERT paper](#)

Demo Time

BGE-M3

- **Multilingual**
 - Supports multiple natural languages
 - Cross-lingual in addition to multi-lingual
- **Multifunctional**
 - Supports both dense and sparse (splade-like) vectors
 - Late interaction model i.e. CoBERT
- **Multigranular**
 - Embeds short phrases as well as long documents
 - Up to 8192 token length

BGE-M3's Sparse Vectors

- **Lexical Retrieval.** The output embeddings are also used to estimate the importance of each term to facilitate lexical retrieval. For each term t within the query (a term is corresponding to a token in our work), the term weight is computed as $w_{qt} \leftarrow \text{Relu}(\mathbf{W}_{lex}^T \mathbf{H}_q[i])$, where $\mathbf{W}_{lex} \in \mathcal{R}^{d \times 1}$ is the matrix mapping the hidden state to a float number. If a term t appears multiple times in the query, we only retain its max weight. We use the same way to compute the weight of each term in the passage. Based on the estimation term weights, the relevance score between query and passage is computed by the joint importance of the co-existed terms (denoted as $q \cap p$) within the query and passage: $s_{lex} \leftarrow \sum_{t \in q \cap p} (w_{qt} * w_{pt})$.

Source: [BGE-M3 paper](#)