ments among all documents in a collection, rather than just among a query-dependent sample; these systems represent a new type of retrieval approaches called *dense retrieval* systems. Thus far, two different families of representations have emerged in dense retrieval. In *single representations* systems, queries and documents are represented with a single embedding, as discussed in Section 3.1. In *multiple representations* systems, queries and/or documents are represented with more than a single embedding, as discussed in Section 3.2. Section 3.3 discusses how representation-focused systems are fine-tuned, exploiting noise-contrastive estimation.

3.1 Single Representations

In interaction-focused systems discussed in Section 2, the query and document texts are concatenated together before processing with sequence-to-sequence models, yielding rich interactions between the query context and the document context, as every word in the document can attend to every word in the query, and vice-versa. At query processing time, every document must be concatenated with the query and must be processed with a forward pass of the whole sequence-to-sequence model. Even if some techniques such as the pre-computation of some internal representations have been proposed [MacAvaney et al. 2020b], interaction-focused systems cannot scale to a large amount of documents. In fact, on a standard CPU, the processing of a query over the whole document collection exploiting an inverted index requires a few milliseconds [Tonellotto et al. 2018], while computing the relevance score of a single query-document pair with a transformer model may requires a few seconds [MacAvaney et al. 2019].

Instead of leveraging sequence-to-sequence models to compute a semantically richer but computationally expensive interaction representation $\eta(q, d)$, representation-focused systems employ encoder-only models to independently compute query representations $\phi(q)$ and document representations $\psi(d)$ in the same latent vector space [Urbanek et al. 2019], as illustrated in Figure 6. Next, the relevance score between the representations is computed via an aggregation function f between these representations:

$$\begin{aligned} \phi(q) &= [\phi_{\texttt{[CLS]}}, \phi_1, \ldots, \phi_{|q|}] = \mathsf{Encoder}(q) \\ \psi(d) &= [\psi_{\texttt{[CLS]}}, \psi_1, \ldots, \psi_{|d|}] = \mathsf{Encoder}(d) \\ s(q, d) &= f\big(\phi(q), \psi(d)\big) \end{aligned} \tag{8}$$

In neural IR, the representation functions ϕ and ψ are computed through fine-tuned encoder-only sequence-to-sequence models such as BERT. The same neural model is

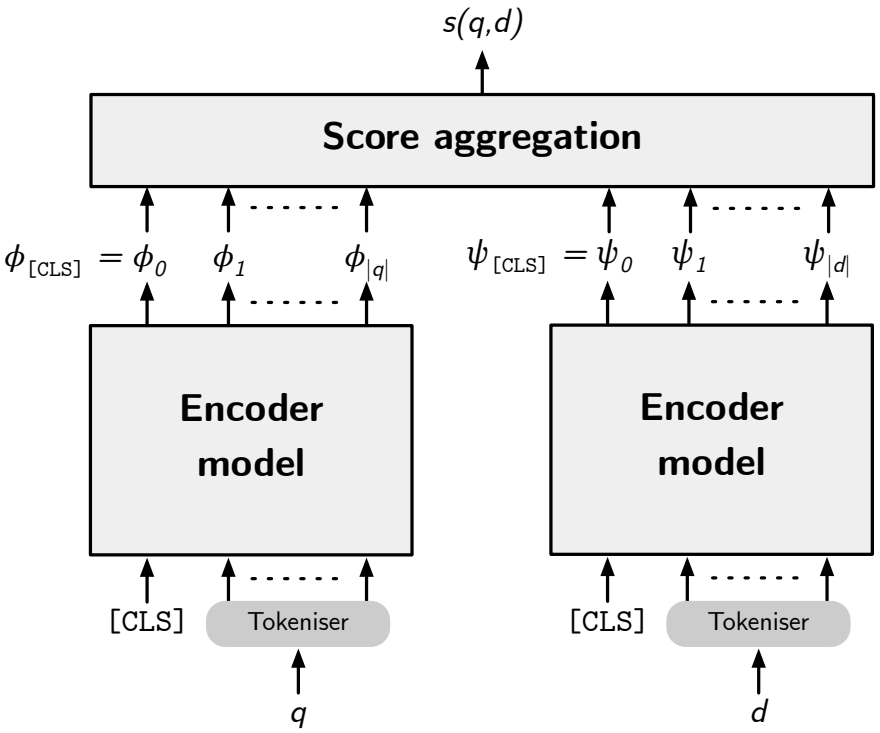

Figure 6: Representation-focused system.

used to compute both the query and the document representations, so the model is also called *dual encoder* or *bi-encoder* [Bromley et al. 1993]. A bi-encoder maps queries and documents in the same vector space $\mathbb{R}^{\ell}$, in such a way that the representations can be mathematically manipulated. Usually, the output embedding corresponding to the `[CLS]` token is assumed to be the representation of a given input text. Using these *single representations*, the score aggregation function is the dot product:

$$s(q,d) = \phi_{[CLS]} \cdot \psi_{[CLS]} \tag{9}$$

Different single-representation systems have been proposed: DPR [Karpukhin et al. 2020], ANCE [Xiong et al. 2021], and STAR [Zhan et al. 2021b] being the most widely adopted. The main difference among these systems is how the fine-tuning of the BERT model is carried out, as discussed in Section 3.3.

3.2 Multiple Representations

Up so far, we considered representation-focused systems in which queries and documents are represented through a single embedding in the latent vector space. This single representation is assumed to incorporate the meaning of an entire text within that single embedding.

In contrast, *multiple representation* systems such as poly-encoders [Humeau et al. 2019], ME-BERT [Luan et al. 2021], ColBERT [Khattab and Zaharia 2020] and COIL [Gao et al. 2021] exploit more than a single embedding to represent a given text, which may allow a richer semantic representation of the content.
Instead of using just the first output embedding $\psi_{\texttt{[CLS]}} = \psi_0$ to encode a document d, poly-encoders [Humeau et al. 2019] exploit the first m output embeddings $\psi_0, \psi_1, \ldots, \psi_{m-1}$. A query q is still represented with the single embedding $\phi_{\texttt{[CLS]}} = \phi_0$, while we need to aggregate the m output document embeddings into a single representation ψ_* to compute the final relevance score using the dot product with the output query embedding. To do so, poly-encoders first compute the m similarities $s_0, \ldots, s_{m-1}$ between the query embedding and the first m document embedding using the dot product. These similarities are transformed into normalised weights $v_0, \ldots, v_{m-1}$ using a softmax operation, and the weighted output embeddings are summed up to compute the final document embedding ψ_* used to compute the relevance score:

$$\begin{aligned}
[\phi_0, \phi_1, \ldots] &= \mathsf{Encoder}(q) \\
[\psi_0, \psi_1, \ldots] &= \mathsf{Encoder}(d) \\
[s_0, s_1, \ldots, s_{m-1}] &= [\phi_0 \cdot \psi_0, \phi_0 \cdot \psi_1, \ldots, \phi_0 \cdot \psi_{m-1}] \\
[v_0, v_1, \ldots, v_{m-1}] &= \mathsf{softmax}([s_0, s_1, \ldots, s_{m-1}]) \\
\psi_* &= \sum_{i=0}^{m-1} v_i \psi_i \\
s(q, d) &= \phi_0 \cdot \psi_*
\end{aligned} \tag{10}$$

Similarly to poly-encoders, ME-BERT [Luan et al. 2021] exploits the first m output embeddings to represent a document d (including the `[CLS]` embedding), but uses a different strategy to compute the relevance score $s(q, d)$ w.r.t. a query q. ME-BERT computes the similarity between the query embedding and the first m document embedding using the dot product, and the maximum similarity, also called *maximum inner product*, represents the relevance score:

$$\begin{aligned}
[\phi_0, \phi_1, \ldots] &= \mathsf{Encoder}(q) \\
[\psi_0, \psi_1, \ldots] &= \mathsf{Encoder}(d) \\
s(q, d) &= \max_{i=0,\ldots,m-1} \phi_0 \cdot \psi_i
\end{aligned} \tag{11}$$

This relevance scoring function, called *max similarity* or *maxsim*, allows us to exploit efficient implementations of maximum inner product search systems, discussed in Section 4. On the contrary, the relevance scoring function in Eq. (10), based on a softmax

operation, does not permit to decompose the relevance scoring to a maximum computation over dot products.

Differently from poly-encoders and ME-BERT, ColBERT [Khattab and Zaharia 2020] does not limit to m the number of embeddings used to represent a document. Instead, it uses all the $1 + |d|$ output embeddings to represent a document, i.e., one output embedding per document token, including the `[CLS]` special token. Moreover, also a query q is represented with multiple $1 + |q|$ output embeddings, i.e., one output embedding per query token, including the `[CLS]` special token. As in other representation-focused systems, query token embeddings are computed at query processing time; queries may also be augmented with additional *masked tokens* to provide "*a soft, differentiable mechanism for learning to expand queries with new terms or to re-weigh existing terms based on their importance for matching the query*" [Khattab and Zaharia 2020]. In current practice, queries are augmented up to 32 query token embeddings. Without loss of generality, query and documents embeddings can be projected in a smaller latent vector space through a learned weight matrix $W \in \mathbb{R}^{\ell' \times \ell}$, with $\ell' < \ell$.

Since there are multiple query embeddings, ColBERT exploits a modified version of the relevance scoring function in Eq (11), where every query embedding contributes to the final relevance score by summing up its maximum dot product value w.r.t. every document embeddings:

$$\begin{aligned} [\phi_0, \phi_1, \ldots] &= \mathsf{Encoder}(q) \\ [\psi_0, \psi_1, \ldots] &= \mathsf{Encoder}(d) \\ s(q,d) &= \sum_{i=0}^{|q|} \max_{j=0,\ldots,|d|} \phi_i \cdot \psi_j \end{aligned} \tag{12}$$

ColBERT's late interaction scoring in Eq. (12), also called *sum maxsim*, performs an all-to-all computation: each query embedding, including the masked tokens' embeddings, is dot-multiplied with every document embedding, and then the maximum computed dot products for each query embedding are summed up. In doing so, a query term can contribute to the final scoring by (maximally) matching a different lexical token. A different approach is proposed by the COIL system [Gao et al. 2021]. In COIL, the query and document `[CLS]` embeddings are linearly projected with a learned matrix $W_C \in \mathbb{R}^{\ell \times \ell}$. The embeddings corresponding to normal query and document tokens are projected into a smaller vector space with dimension $\ell' < \ell$, using another learned matrix $W_T \in \mathbb{R}^{\ell' \times \ell}$. Typical values for ℓ' range from 8 to 32.

The query-document relevance score is the sum of two components. The first compo-

nent is the dot product of the projected query and document [CLS] embeddings, and the second component is the sum of sub-components, one per query token. Each sub component is the maximum inner product between a query token and the document embeddings for the same token:

$$\begin{aligned}
[\phi_0, \phi_1, \ldots] &= \mathsf{Encoder}(q) \\
[\psi_0, \psi_1, \ldots] &= \mathsf{Encoder}(d) \\
[\phi'_0, \phi'_1, \ldots] &= [W_C\phi_0, W_T\phi_1, \ldots] \\
[\psi'_0, \psi'_1, \ldots] &= [W_C\psi_0, W_T\psi_1, \ldots] \\
s(q,d) &= \phi'_0 \cdot \psi'_0 + \sum_{t_i \in q} \max_{t_j \in d,\, t_j = t_i} \phi'_i \cdot \psi'_j
\end{aligned} \tag{13}$$

The COIL's scoring function, based on lexical matching between query and document tokens, allows us to pre-compute the projected document embeddings and, for each term in the vocabulary, to concatenate together the embeddings in the same document and in the whole collection, organising them in posting lists of embeddings, including a special posting list for the [CLS] token and its document embeddings. This organisation permits the efficient processing of posting lists at query time with optimised linear algebra libraries such as BLAS [Blackford et al. 2002]. Note that the projected query embeddings are still computed at query processing time.

3.3 Fine-tuning Representation-focused Systems

The fine-tuning of a bi-encoder corresponds to learning an appropriate inner-product function suitable for the ad-hoc ranking task, i.e., for relevance scoring. As in Section 2.5, we have a neural IR model $\mathcal{M}(\theta)$, parametrised by θ, that computes a score $s_\theta(q,d)$ for a document d w.r.t. a query q. We now frame the learning problem as a probability estimation problem. To this end, we turn the scoring function into a proper conditional distribution by using a softmax operation:

$$p_\theta(d|q) = \frac{\exp\left(s_\theta(q,d)\right)}{\sum_{d' \in \mathcal{D}} \exp\left(s_\theta(q,d')\right)} \tag{14}$$

where $p_\theta(d|q)$ represents the posterior probability of the document being relevant given the query. We assume to have a joint distribution p over $\mathcal{D} \times \mathcal{Q}$, and we want to find the parameters θ^* that minimise the cross entropy l_{CE} between the actual proba-

bility $p(d|q)$ and the model probability $p_\theta(d|q)$:

$$\theta^* = \arg\min_\theta \mathbb{E}\Big[l_{CE}(d,q)\Big] = \arg\min_\theta \mathbb{E}\Big[-\log\big(p_\theta(d|q)\big)\Big] \tag{15}$$

where the expectation is computed over $(d,q) \sim p$. If the scoring function $s_\theta(q,d)$ is expressive enough, then, for some θ, we have $p(d|q) = p_\theta(d|q)$.
The cross entropy loss in Eq. (15) is difficult to optimise, since the number of documents in $\mathcal{D}$ is large, and then the denominator in Eq. (14), also known as *partition function*, is expensive to compute. In *noise contrastive estimation* we choose an artificial noise distribution g over $\mathcal{D}$ of *negative samples* and maximise the likelihood of $p_\theta(d|q)$ contrasting $g(d)$. Given $k \geq 2$ documents $\mathcal{D}_k = \{d_1, \ldots, d_k\}$, for each of them we define the following conditional distribution:

$$\hat{p}_\theta(d_i|q,\mathcal{D}_k) = \frac{\exp\big(s_\theta(q,d_i)\big)}{\sum_{d'\in\mathcal{D}_k}\exp\big(s_\theta(q,d')\big)} \tag{16}$$

which is significantly cheaper to compute that Eq. (14) if $k \ll |\mathcal{D}|$. Now, we want to find the parameters $\theta^\dagger$ that minimise the noise contrastive estimation loss l_{NCE}, defined as:

$$\theta^\dagger = \arg\min_\theta \mathbb{E}\Big[l_{NCE}(\mathcal{D}_k,q)\Big] = \arg\min_\theta \mathbb{E}\Big[-\log\big(\hat{p}_\theta(d_1|q,\mathcal{D}_k)\big)\Big] \tag{17}$$

where the expectation is computed over $(d_1,q) \sim p$ and $d_i \sim g$ for $i = 2,\ldots,k$.
The end goal of this fine-tuning is to learn a latent vector space for query and document representations where a query and its relevant document(s) are closer, w.r.t. the dot product, than the query and its non-relevant documents [Karpukhin et al. 2020]; this fine-tuning approach is also called *contrastive learning* [Huang et al. 2013].
Negative samples are drawn from the noise distribution g over $\mathcal{D}$. In the following, we list some negative sampling strategies adopted in neural IR.

- *Random sampling*: any random document from the corpus is considered non-relevant, with equal probability, i.e., $q(d) = 1/|\mathcal{D}|$. Any number of negative documents can be sampled. Intuitively, it is reasonable to expect that a randomly-sampled document will obtain a relevance score definitely smaller than the relevance score of a positive document, with a corresponding loss value close to 0. Negative documents with near zero loss contribute little to the training convergence to identify the parameters $\theta^\dagger$ [Johnson and Guestrin 2018, Katharopoulos and Fleuret 2018].