Supervised Information Retrieval (Chapter 15)

Definition 1 (Encoder)
Given a set of documents $D_{1..|D|}$ and queries $Q_{1..|Q|}$, an ***Encoder model*** *$\mathcal{M}$ maps text $t_i \in D_{1..|D|} \cap Q_{1..|Q|}$ into an* ***embedding****, i.e. a fixed-size vector v_i such that $\mathcal{M}(Q_j) \sim \mathcal{M}(D_k)$* ***if*** *the document D_i is* ***relevant*** *for Q_j, and $\mathcal{M}(Q_j) \not\sim \mathcal{M}(D_k)$ otherwise.*

Definition 2 (Dense retrieval)
An Encoder model $\mathcal{M}$ can be used for Information Retrieval by following these steps:

1. *On initialisation, index the search collection of documents D by encoding each document D_i in the collection with $\mathcal{M}$: $v_i = \mathcal{M}(D_i)$ and persist the representations of all documents*

2. *On search, encode the given query Q_j to $v_j = \mathcal{M}(Q_j)$ and* ***rank*** *the retrieved list of documents by the ascending distance of v_j to the indexed representations of D.*

When using neural language models, such as Transformers, the encoded representations v comprise a few hundred float numbers, as compared to tens of thousands when we use exact-match TF-IDF representations; Hence the label of Dense retrieval.

Definition 3 (Encoder training objective)
Let $D_{1..|D|}$ denote a set of documents, $Q_{1..|Q|}$ a set of queries and $J_{1..|J|}(Q_j, D_i)$ a set of relevance judgements marking pairs of documents D_i relevant for query Q_i.

The training objective of an Encoder model $\mathcal{M}$ is then to ***minimise*** *the distance of $\mathcal{M}(Q_j)$ to $\mathcal{M}(D_i)$, while* ***maximising*** *the distance of $\mathcal{M}(Q_j)$ to the representations $\mathcal{M}(D_k)$ of irrelevant documents.*

Following are some formulations of Encoder training objectives (i.e. losses $\mathcal{L}$) that are being used for training Encoder for Information Retrieval:

- ***Contrastive objectives*** *directly train the Encoder to minimise the distance between the query representation and its relevant documents D_{rel}:*

$$\mathcal{L}_{contrastive}(Q_j, D_{rel}) = min\Big[dist\Big(\mathcal{M}(Q_j), \mathcal{M}(D_{rel})\Big)\Big]$$

- ***Margin objectives*** *train the Encoder to maximise the difference (i.e. margin) between the query's distance to (i) a relevant document D_{rel} and (ii) not relevant document D_{nrel}:*

$$\mathcal{L}_{margin}(Q_j, D_{rel}, D_{nrel}) = max\Big[dist\Big(\mathcal{M}(Q_j), \mathcal{M}(D_{rel})\Big) - dist\Big(\mathcal{M}(Q_j), \mathcal{M}(D_{nrel})\Big)\Big]$$

  This objective can also be formulated such that the value of the difference approaches a specific value V:

$$min\Big[\Big|V - \Big(dist\Big(\mathcal{M}(Q_j), \mathcal{M}(D_{rel})\Big) - \ dist\Big(\mathcal{M}(Q_j), \mathcal{M}(D_{nrel})\Big)\Big)\Big|\Big]$$

- ***Batch negatives objectives*** *aggregate the results of other objective(s) for a chosen query Q_j and a relevant document D_{rel} over* ***multiple non-relevant documents*** *D_{nrel}:*

$$\mathcal{L}_{batch}(Q_j, D_{rel}, D_{nbatch}) = opt\Big[\frac{1}{|D_{nbatch}|} \sum_{D_{nrel} \in D_{nbatch}} \Big(\mathcal{L}_{any}(Q_j, D_{rel}, D_{nrel})\Big)\Big]$$

For more examples and ready-to-use implementations ***take a look in the attached Google Colab*** *for this week and into* ***Sentence Transformers documentation:*** *`https://www.sbert.net/docs/package_reference/losses.html`*

Definition 4 (Learning to rank IR System (Reranker))
Let's have a set of documents $D_{1..|D|}$, a set of queries $Q_{1..|Q|}$ and a set of relevance judgements $J_{1..|J|}(Q_j, D_i)$ marking pairs of documents D_i relevant for query Q_i.

Then, the objective of a Reranker $\mathcal{R}$ is to ***adjust the ranking*** *given by a Dense retrieval using Encoder, such that the top-n results of the IR system include* ***more*** *relevant documents than without adjusting the ranking.*

Usually, $\mathcal{R}$ is used to re-rank the responses of fast Dense retrieval IR System by (i) computing the relevance score of each document in top-n results and (ii) reordering the documents by the $\mathcal{R}$'s score:

$$ordering(D_i, D_k) = \begin{cases} D_i > D_k, & \textit{if } \mathcal{R}(Q_i, D_i) > \mathcal{R}(Q_i, D_k) \\ D_i < D_k, & \textit{if } \mathcal{R}(Q_i, D_i) < \mathcal{R}(Q_i, D_k) \end{cases}$$

By re-ordering top-n results previously ranked by Encoder, we can improve the quality of sole Dense retrieval using a more expressive but slower model $\mathcal{R}$.

Exercise 15/1

Consider a task of Information retrieval in a community question-answering forum such as CQADubStack, where we have **questions** associated with the following attributes: (i) **List of answers**, (ii) **upvotes**, (iii) **links to related questions**, (iv) **users' reputation**.

a) How would you formulate the optimisation problem using this dataset such that the resulting Encoder model can be used to search for an answer to new queries?

b) Which of the objectives from Definition 3 can be applied in our scenario?

c) In addition to the main objective, can we use additional data from our dataset to make Encoder's representations more accurate for our Information retrieval task? How would you formulate the optimisation problem using multiple objectives?

d) Take a look at the Google Colab examples for this week. How would you apply these examples to the aforementioned task?

---

a) We want to optimise Encoder to return similar representations to pairs of Query+Answer that are mutually relevant. At the same time, we want the Encoder to return diverse representations for both the irrelevant answers and answers with a low number of upvotes.

b) All of them. E.g. we can pick a true answer and a random answer to a given query for Contrastive objectives. Or using best and worst answers to a given question as positives and negatives to the Margin objective.

c) Yes. Together with the Query+Answer optimisation, we can minimise the mutual distance of Related questions or multiple highly-voted answers. We can make multiple filters of otherwise randomly-sampled negative answers: e.g. we can assume that popular questions with high lexical overlap would be linked if they were mutually relevant, and if they are not, they might be used as soft negatives.

d) Discuss and check that the students know how to use these objectives.

Exercise 15/2

Look at the definition 3 and try to compare the **Contrastive** and **Margin** objective in the data settings introduced in Exercise 15/1. What are the shortcomings and benefits of each of these objectives over each other?

- Contrastive objective can not utilize negative examples.
- Margin objective can utilize ground-truth distance evaluations, such as upvotes
- Margin objective is partially tolerant to noisy annotations of relevant documents

Exercise 15/3

Look at the definition of **Batch negatives objectives** in Definition 3 and think about the benefits of such formulation in practice. What are the main benefits of this approach compared to single-negative objectives?

Hint: think about the situation where you do not have the annotations of ***hard negatives****, where you are sure that answers are not relevant for a given question.*

Batch negatives approach can be used if we do not have hard negatives, given its robustness to specific samples; If suffices if we can assume that most of our negatives are irrelevant.

Given this assumption, we can "challenge" Encoder with seemingly-negative answers (e.g. having large lexical overlap) and train it to infer more robust representations.

Exercise 15/4

Consider a pairwise Reranker $\mathcal{R}$ from Definition 4, that reranks top-100 search results of Dense retrieval.

a) How many predictions of Encoder $\mathcal{M}$ and Reranker $\mathcal{R}$ must we perform when processing a single query?

b) In a common scenario, we can use BERT-based models for both Encoder and Reranker. On a single CPU, the prediction of a small BERT model takes around 2 seconds per prediction. How long will it take to process a single query using a BERT-based Encoder and Reranker? How will this time change, if we use GPUs, where the same inference takes 0.1 seconds in average?

You can consult the example implementation of Retrieve+Re-Rank Information retrieval system that you can find in the attached Google Colab for this week.

a) 1+100=101. **b)** 2 seconds $\times$ 101 inferences = 202 seconds = 3 minutes and 22 seconds, or 10.1 seconds on GPU.

Exercise 15/5

Does every Reranker improve the quality of search results of the enclosing IR System? What properties should the Reranker have to maximise its benefit for the quality of the search results? Discuss.

No, Reranker can also make mix up the results in undesired way.
In the ideal scenario, Reranker fixes specific problems of the Encoder: For instance, it reranks the documents robustly to the false exact matches, or it can verify that the documents (i.e. answers) truly belong to the topic of the query. The best but also most elaborate strategy is to refine the Reranker by the observation of the Encoder-only results.