Implementation of a simple pairwise Machine-Learned Ranking algorithm using SVMs

Carlos Morote García

David Lorenzo Alfaro

Universidad Politécnica de Madrid
ETSIINF

April, 2022

Contents

1 Introduction

Learning to rank for information retrieval, IR in short, is a task to automatically construct a ranking model using training data, such that the model can sort new objects according to their degrees of relevance, preference, or importance. As this IR become more complex, learning to rank approaches are being developed to automatically tune their parameters. Using online learning to rank, retrieval systems can learn directly from implicit feedback inferred from user interactions. (Hofmann, Whiteson, & de Rijke, 2013; T.-Y. Liu et al., 2009)

A particularly successful approach shown to comply with those pressing needs effectively is machine-learned ranking (MLR), which is aimed at utilizing machine learning algorithms for information retrieval. MLR techniques are typically divided into three categories, accounting for the type of input to the ranking models. Namely, the pointwise approach yields per-document marginal relevance scores, whereas the pairwise and listwise approaches account for the relative relevance of the query with respect to a set of documents (pairs of documents, in the case of pairwise ranking) to produce a more informed, yet not necessarily more meaningful, ordering.

In this work we describe a basic implementation of a retrieval function via linear support-vector machines, following the framework proposed in (Joachims, 2002), which lies within the pairwise ranking paradigm. To that end, we utilize a small sample of data from the *LOINC* database to extract a meaningful set of document descriptors both at training and inference time. The whole project are located in github and can be access through this link[1].

2 Retrieval function for pairwise ranking

One of the main problems in machine-learned ranking is the determination of preference relations between interested units. As aforementioned, in this context ranking can be defined as learning a function with the ability to organize units according to a given preference relation. This type of problem is often treated as a classification problem whereby examples are composed of pairs of samples, i.e., following a pairwise approach for ranking. The paradigm is changed from the pointwise or listwise approaches from ranking to classifing tuples of documents to learn their relative order. (Üniversitesi & İstatistik Bölümü, 2018)

Following the notation in (Joachims, 2002), the problem of information retrieval can be formalized as follows. For a query q and a document collection $D = \{d_1, \ldots, d_m\}$, the optimal retrieval system should return a ranking r^* that orders the documents in D according to their relevance to the query. In this work, queries are represented as a set of keywords, expressed in natural language, whereas documents, i.e., observations, are represented as a set of characteristics which vary in nature. Both queries and documents should be subsequently transformed into the appropriate feature space for machine learning models to be able to handle them in a meaningful fashion, i.e., information should be properly encoded for models to learn how to map input samples with their relevance.

Formally, the ML model is targeted at learning a ranking function, $r_{f(q)}$ which approx-

[1] `https://github.com/CarlosMorote/RankingSVM`

imates the optimum, r^*. If a document d_i is ranked higher than d_j for an ordering r, i.e. $d_i <_r d_j$, then $(d_i, d_j) \in r$, otherwise $(d_i, d_j) \notin r$. Assuming the ranking function to be a strict ordering, for all pairs $(d_i, d_j) \in D \times D$, either $d_i <_r d_j$ or $d_j <_r d_i$.

Consequently, obtaining an arbitrary ordering r is straightforward upon yielding all ranks for the pairwise documents bijections, i.e., for each permutation of $(d_i, d_j) \in D \times D$, such that $i \neq j$. This is known as the pairwise transform which, as proven in (Herbrich, Graepel, & Obermayer, 1999), is able to map an ordinal regression problem into a binary classification problem by computing the difference of all comparable elements such that the original data is transformed into $(x_k^{'}, y_k^{'}) = (x_i - x_j, sign(y_i - y_j))$. The approach is, indeed, based on a mapping from objects to scalar utility values.

In particular, each document in each tuple is first embedded into a feature space, via a function $g : D \to R^T$, where T denotes the cardinality of the feature space, and then a ranking function, outputs the relative relevance of the pair of documents, $h : R^T \times R^T \to Y \in \{-1, 1\}$. Note that, for simplicity, we do not include the query anywhere in function g, albeit the representation of documents in the feature space is not independent of the query, as we will discuss in the following sections.

Consequently we can derive that, given an arbitrary query q_i, a pair of documents, $(d_i, d_j) \in D \times D$ and an ordering function $r_{f(q_i)}$, $(g(d_i), g(d_j), 1) \iff d_i < r_{f(q_i)}\, d_j$.

2.1 Ranking SVM algorithm

We are aimed at utilizing a support-vector machine to obtain a ranking function $f_{\vec{w}}$ that generalizes well beyond training data. The main rationale is that such learned retrieval function can be represented as a linear combination of the feature vectors, $\Phi(q, d)$, and $\vec{(}w)$ which is a weight vector that is adjusted by learning, such that:

$$\begin{aligned}(d_i, d_j) \in f_{\vec{w^*}}(q) \\ &\iff \vec{w^*}\Phi(q, d_i) > \vec{w^*}\Phi(q, d_j) > \\ &\iff \sum \alpha^*_{k,l}\Phi(q_k, d_l)\Phi(q, d_i) > \alpha^*_{k,l}\Phi(q_k, d_l)\Phi(q, d_j)\end{aligned} \tag{1}$$

Where the $\alpha^*_{k,l}$ can be derived from the values of the dual variables at the solution. In our approach, alpha values can be ignored because we are utilizing linear kernels.

Likewise, since the ranking function, i.e., $f_{\vec{w}}$ will be used to rank documents according to new queries q. To that end, it is sufficient to sort the documents by their value of

$$rsv(q, d_i) = \vec{w^*}\Phi(q, d_i) = \sum \alpha^*_{k,l}\Phi(q_k, d_l)\Phi(q, d_j) \tag{2}$$

3 Implementation

In this section we discuss how the algorithm of the paper proposed by (Joachims, 2002) was implemented. We note how we have linked this theoretical proposal with a simple Python implementation that in turn will allow us to generalize it for as many queries as we want or train the ranking with as many documents as desired via a common interface.

Our implementation divides the information retrieval problem into two main subtasks. The first is aimed at providing a general interface able to model and encode input data in a flexible fashion for it to be fed into a SVM model for both training and inference. The second is in charge of providing the directives to train and to make predictions with the model (i.e., generate meaningful orderings on unseen queries).

3.1 LOINC

As stated in (McDonald et al., 2003), the Logical Observation Identifier Names and Codes (LOINC) database provides a universal code system for reporting laboratory and other clinical observations. Its purpose is to identify observations in electronic messages such as Health Level Seven observation messages.

Part of the data was originally given to us by the instructors, albeit it can be as well yielded using the *SearchLOINC*[2] tool. We also employ *SearchLOINC* to enrich our initial dataset, as we will discuss in the next section.

3.2 Dataset

We start with a fundamental part of any machine learning model: the data. Before going on to explain how it has been coded and the flow designed for such processing, we must explain some restrictions onto how the data must be structured. Following the rationale behind relational database management systems, which attempt to reduce redundancies effectively, we separate data into three subsets (see Figure 1). One models the collection of laboratory observations, each characterized by a set of features (see Table 1). On the other hand, queries are encoded in a different module, and each is assigned a unique identifier. The last document models per-query-per-document relations, entailing relating their respective identifiers, along with the ranking of the i^{th} document for the j^{th} query. Note that the last two files will only be necessary in the learning phase. In this scenario we propose and set as a restriction that only the documents in the file as such can be ranked.

Moreover, we enrich the initial available information by including further document descriptors. That is, a series of new features were (manually) integrated, yielding them directly from LOINC. These new columns are (1) *class*, (2) *discourage* and (3) *rank_loinc*, the latter referring to the internal LOINC ranking as a document of generic importance. That notwithstanding, we would like to place special emphasis that, rather than building a training data similar to that of a real scenario, this work is primarily concerned with providing a general framework for (i) storing observations, queries and rankings, (ii) properly encoding the data for ML models to handle it, and (iii) produce orderings on new queries. The main rationale behind us not focusing on extending the dataset is that we do not have at our disposal actually informed per-query-per-document judgments, other than, for example, utilizing LOINC's ranking on a set of queries as ground truth to learn the retrieval function. Thus, in presence of such data, be it a set of relevance judgments, click-through data or any of the likes, extending the dataset and building new models is straightforward.

[2] `https://loinc.org/search-app/`

3.3 Feature mapping

So far, we have mentioned how it is necessary to map documents into a set of salient features. In the end, the multivariate relations among those features shall explain, to a greater extent than random, how a document may or may not be relevant to a query. The design of a feature mapping function shall effectively describe and capture the extent to which a query and a document match. Formally, what we are to describe is the feature mapping function, $\Phi(q, d)$, formerly defined as g for the sake of simplicity. Note that there is no explicit representation for queries, and neither is for documents, albeit there is some query-invariant document features.

Strictly categorical features, namely, *system, property, scale, class* and *discourage* are one-hot encoded. On the other hand, null values in the feature *rank* are replaced by an arbitrarily large negative value (our particular take was -999999999) to signify strong opposition with non-null values, i.e., with lab tests that are of special relevance.

When it comes to features *long_common_name* and *component*, we compute their textual similarity with respect to the query. To that end, we encode both the query and the document textual properties into the embedding space. These embeddings are computing utilizing Sentence Transformers. Formerly known as Sentence-BERT or SBERT, and introduced in (Reimers & Gurevych, 2019) proposes a modification of the BERT (Devlin, Chang, Lee, & Toutanova, 2018) network able to derive semantically meaningful sequence embeddings that can be compared by means of similarity measures like cosine-similarity or the Euclidean or Manhattan distance. Essentially, they fine-tune BERT general-purpose embeddings on different Semantic Textual Similarity datasets using siamese neural networks and append an extra pooling layer in order for the model to create fixed-size representations for input sentences of varying lengths.

In our work, we use the *S-Biomed-Roberta-snli-multinli-stsb*[3] pretrained sentence transformer. This model fine-tunes the *BioMed-RoBERTa-base*[4] (Gururangan et al., 2020) pretrained model, on several on different Semantic Textual Similarity (STS) datasets, e.g., the Stanford Natural Language Inference (SNLI) corpus (Bowman, Angeli, Potts, & Manning, 2015). Likewise, the *BioMed-RoBERTa-base* pretrained model is a language model based on the *RoBERTa-base* (Y. Liu et al., 2019) architecture, fine-tuned on 2.68 million scientific papers from the S2ORC (Lo, Wang, Neumann, Kinney, & Weld, 2019) corpus via continued pretraining.

The idea is that, following a feature-based approach, semantically similarities between the semantic representations of features *long_common_name* and *component* and that of the query may effectively help disambiguate query-document relevance in a more meaningful fashion than query-invariant document features, i.e., it is very unlikely that a document having a certain feature is relevant, no matter what the query is.

Lastly, nearly uniformly distributed random variables (e.g., identifiers) are disregarded, since they convey no discriminatory power.

[3]Model card: `https://huggingface.co/pritamdeka/S-Biomed-Roberta-snli-multinli-stsb`
[4]Model card: `https://huggingface.co/allenai/biomed_roberta_base`

3.4 Ranking

Once we have transformed our information to a new space according to $\Phi(q, d_i)$ we can proceed to analyze the simple flow that has been made for the implementation of the ranking by means of a support vector machine. The model will learn the partial orderings via pairwise transformation, i.e., for each pair of input samples $x_i, x_j \in R^T \times R^T$, $h : R^T \times R^T \to Y \in \{-1, 1\}$, where the input will consist of $(x_i - x_j)$ and the out will be $sign(y_i - y_j))$. This data is subsequently feed into a SVM to feed the MLR model, and to obtain a ranking function $f_{\vec{w}^*}$, as discussed in Section 2.

After training this SVM we can extract the weights learned by the model, $\vec{w}$, to perform inference. That is, given a new query we compute the set of document properties and multiply the learned weights against the properties (see Equation 2). In this way we obtain the ranking position for each document. With this it would only remain to sort them accordingly.

It is worth noting that the work by Joachims emphasizes the use of SVMs as ranking function algorithms where, in absence of ground truth, ranks are inferred from click-through data at training. As long as, we do not hold such information within our data we cannot use it. Therefore, we **assume** that the **order of the original documents in each query as its ranking values**. Because such orderings do not seem to be correct (and far from optimal), the training procedure is necessarily misleading. Either way, be the ranking based on relevance judgements or in click-through data, the feature mapping and training processes would remain the same, being only necessary to tune the values for the per-query-per-document relation, which should then encode, for example, the number of clicks, suggesting its relative relevance.

The performance of the model it is as expected with the dataset we worked with. We only had three queries availble with its corresponding ranking. This is not close enough to be sufficient data to get a model which is able to generalize properly. Moreover, we assumed the ranking is the order of appearance. So, it is likely that might be a better ordering to such queries.

References

Bowman, S. R., Angeli, G., Potts, C., & Manning, C. D. (2015, 8). A large annotated corpus for learning natural language inference. In (p. 632-642). Association for Computational Linguistics (ACL). Retrieved from `http://aclweb.org/aclwiki/index.php?` doi: 10.18653/v1/d15-1075

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2018, 10). Bert: Pre-training of deep bidirectional transformers for language understanding. *NAACL HLT 2019 - 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference*, *1*, 4171-4186. Retrieved from `http://arxiv.org/abs/1810.04805`

Gururangan, S., Marasović, A., Swayamdipta, S., Lo, K., Beltagy, I., Downey, D., & Smith, N. A. (2020). Don't stop pretraining: adapt language models to domains and tasks. *arXiv preprint arXiv:2004.10964*.

Herbrich, R., Graepel, T., & Obermayer, K. (1999). Support vector learning for ordinal regression. *IEE Conference Publication*, *1*, 97-102. doi: 10.1049/CP:19991091

Hofmann, K., Whiteson, S., & de Rijke, M. (2013). Balancing exploration and exploitation in listwise and pairwise online learning to rank for information retrieval. *Information Retrieval*, *16*(1), 63–90.

Joachims, T. (2002). Optimizing search engines using clickthrough data. In *Proceedings of the eighth acm sigkdd international conference on knowledge discovery and data mining* (pp. 133–142).

Liu, T.-Y., et al. (2009). Learning to rank for information retrieval. *Foundations and Trends® in Information Retrieval*, *3*(3), 225–331.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., … Stoyanov, V. (2019, 7). Roberta: A robustly optimized bert pretraining approach. *arXiv*. Retrieved from `http://arxiv.org/abs/1907.11692`

Lo, K., Wang, L. L., Neumann, M., Kinney, R., & Weld, D. S. (2019). S2orc: The semantic scholar open research corpus. *arXiv preprint arXiv:1911.02782*.

McDonald, C. J., Huff, S. M., Suico, J. G., Hill, G., Leavelle, D., Aller, R., … others (2003). Loinc, a universal standard for identifying laboratory observations: a 5-year update. *Clinical chemistry*, *49*(4), 624–633.

Reimers, N., & Gurevych, I. (2019, 8). Sentence-bert: Sentence embeddings using siamese bert-networks. *EMNLP-IJCNLP 2019 - 2019 Conference on Empirical Methods in Natural Language Processing and 9th International Joint Conference on Natural Language Processing, Proceedings of the Conference*, 3982-3992. Retrieved from `http://arxiv.org/abs/1908.10084`

Üniversitesi, A. K., & İstatistik Bölümü. (2018). A pairwise ranking algorithm for information retrieval..

A Additional information

loinc_num	Unique identifier per document within LOINC framework
long_common_name	Name / Title which briefly describes in textual manners the document
component	Mainly component of the element modelled by the doc
system	The specimen or thing upon which the observation was made.
property	The characteristic or attribute of the analyte
scale	How the observation value is quantified or expressed: quantitative, ordinal, nominal.
class	Clinical Term Classes
discourage	If the document has been dishearten or not
rank	Models if the doc is in the list which contains just over 2000 LOINC codes that represent about 98% of the test volume carried by three large organizations that mapped all of their lab tests to LOINC codes.

Table 1: LOINC Feaures that describes each document in *docs* file[5]

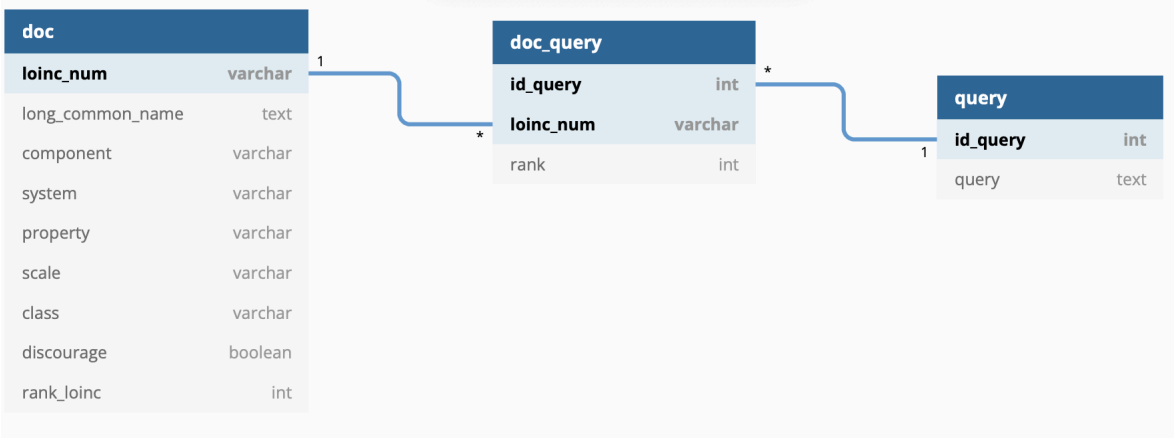

Figure 1: Class diagram which models the relation between entities.