Learning to Rank Arguments with Feature Selection

Notebook for the Touché Lab on Argument Retrieval at CLEF 2021

Christopher Akiki[1], Maik Fröbe[2], Matthias Hagen[2] and Martin Potthast[1]

[1]*Leipzig University*
[2]*Martin-Luther-University Halle-Wittenberg*

Abstract

This notebook documents our participation in the *Argument Retrieval for Controversial Questions* subtask of Touché 2021. We submit five runs tackling argument retrieval with four different methodological paradigms: (1) a Dirichlet-smoothed language-model with filtering of low-quality arguments, (2) two learning to rank approaches using argumentative features, (3) a reranking approach that casts argument-retrieval as a question-answering (QA) task, and (4) a transformer-based query expansion method that enriches the query with topically relevant keywords.

Keywords

argument retrieval, learning to rank, query expansion,

1. Introduction

The web has evolved into a participatory resource, where sharing one's opinion is as easy as the click of a button [1]. Retrieval systems, traditionally optimized for ad hoc retrieval, must increasingly deal with very subjective and nuanced user-generated content while providing users with deliberative information needs with the best arguments supporting or refuting their controversial questions. The args.me dataset [2] forms the testbed of the first Touché subtask on argument retrieval [3] helping researchers explore retrieval systems for controversial questions. The args.me corpus consists of more than 300,000 arguments mined from online debate portals, and ours is the task of retrieving the best—both in relevance and in quality—arguments pertinent to a given controversial question.

The current search landscape is not especially attuned to nuances such as those afforded by argument retrieval, usually preferring to let "the stakeholders compete for what opinion ranks higher" [4]. In this argumentative setting, a single document can no longer satisfy one query, just as surely as debates typically cannot be reduced to one morsel of text [5], but typically oscillate in a dialectical back-and-forth across orthogonal dimensions of quality: logical, rhetorical, and dialectical [6]. A deliberative information need can therefore only be satisfied by a diverse set of documents. The ability to specifically handle the arguments embedded within those documents is an attempt to address that problem using computational argumentation analysis [4, 7, 8].

Having recognized the non-trivial nature of the task and being allowed a maximum of five runs to be submitted for human judgment, we deemed it prudent to diversify our solutions. In

CLEF 2021 – Conference and Labs of the Evaluation Forum, September 21–24, 2021, Bucharest, Romania

so doing, we can judge and compare the merits of each method and better understand where future efforts ought to be invested.

The first run we submit uses Elasticsearch's LM Dirichlet[1] similarity module with default parameter values, that is a language model with Dirichlet smoothing [9] with an additional filtering step that removes low-quality documents from the set of results. Our second and third runs rerank the top-100 results of Elasticsearch's LM Dirichlet similarity with LambdaMART [10] using 4 respectively 9 features. The fourth run we submit relies on the hypothesis that the tasks of question answering and argument retrieval are close enough for a solution of the latter to be based on methods developed for the former. To that end, we rerank the documents retrieved by the first run using the question-answering variant of Google's Universal Sentence Encoder [11, 12]. As fifth and final run, we submit a template-based—in Schick and Schütze's [13] usage—query expansion method that enriches the original topic with a list of thematically related keywords.

2. Learning-based Reranking of Arguments

The five approaches to argument retrieval we study start from first principles. We develop an approach close the baseline DirchletLM model, but filtering low-quality arguments first, employ the well-known learning to rank approach LambdaMART with and without feature selection, and attempt to improve upon our previous approach based on neural query expansion.

2.1. Baseline: Multi-field Dirichlet-LM with Quality Filter

The surprising competitiveness of the baseline argument retrieval system [4] of last year's workshop [14] makes this approach a natural choice for our most basic run. We index both the text field as well as the conclusion field of the args.me corpus using Elasticsearch's probabilistic `DirichletLM` similarity module [9]. At retrieval time, we rely on the default behavior of the `multi_match` query[2] to retrieve the top-1000 most relevant documents to every topic query.

However, from our last year's experience, we recall that the args.me dataset it comparably noisy. To filter out noise from the retrieved documents, we begin by embedding the corpus using the transformer-encoder-based[3] variant Google's Universal Sentence Encoder [11] to embed each of the 387,740 arguments into a 512-dimensional space. Sharing only architectural similarities with BERT [15], the Universal Sentence Encoder uses the rule-based Penn Treebank Tokenizer as well as more explicitly semantically-aware pretraining tasks:[4]

- *Skip-thought* is a self-supervised pretraining task, originally devised to use LSTMs to provide high-quality sentence vectors by training on a large amount of contiguous text [16].
- *Natural language response suggestion* imparts conversational awareness to the sentence encoder, which fits quite well to the task at hand. The goal of this supervised task is to predict the best short reply among millions of options in response to an email [17].

[1] https://www.elastic.co/guide/en/elasticsearch/reference/current/index-modules-similarity.html#lm_dirichlet
[2] https://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl-multi-match-query.html
[3] https://tfhub.dev/google/universal-sentence-encoder-large/5
[4] Both in contrast to BERT's learned subword tokenizer and solely self-supervised pretraining signal

Table 1
Overview of the 31 learning to rank features used in our feature selection.

Argument Text		Discussion		Argument		Query		Other	
Description	Count	Description	Count	Description	Count	Description	Count	Description	Count
BM25	3	Discussion length	1	Premise length	1	Word count	1	Garbage	1
DirichletLM	3	Number of premises	1	Argument position (abs.)	1	Concept count	1		
JelinekMercer	3	Premise length (mean)	1	Argument position (rel.)	1	Entity count	1		
IB	3								
DFI	3								
DFR	3								
Baseline	1								
USE_QA_T_C	1								
USE_QA_T_T	1								

- *Stanford natural language inference* is a labeled dataset [18] of 570,000 sentence pairs. This can be seen as a supervised variant of BERT's next sentence prediction pre-training task. In this instance however, entailment, contradiction, and irrelevance are explicitly labeled in the data itself, rather than implied by the relative position of two sentences in an unlabeled corpus of contiguous text.

We then cluster these 387,740 dense vectors using the `scikit-learn` [19] implementation of k-means with default arguments and number of clusters set to a heuristically chosen value of 100. Proceeding to manually labeling each of the clusters, we observe they carry both syntactic as well as semantic coherence. For this approach, we are interested in those clusters that correspond to noise. They tend to exhibit syntactic coherence, like, for instance, a cluster that consists of YouTube links, or another that consists solely of repeated short idiosyncratic phrases one tends to find on online debate websites (e.g., "I agree."). Any argument belonging to a cluster that was identified as noise is filtered out of this and subsequent runs.

2.2. Learning to Rank: LambdaMART

We submit two learning to rank runs with LambdaMART [10] trained on subsets of 31 features belonging to 5 different feature classes. Table 1 provides an overview of the 31 features grouped by the 5 feature classes. Overall, we implement 21 features that calculate the similarity between (parts of) an argument and the query (labeled as argument text features in Table 1). These 21 argument text features consist of 6 retrieval models implemented in Elasticsearch, for which we produce 3 seperate features per text field available in the args.me corpus (the premise, the conclusion, and the argument's topic), plus the retrieval scores of our 3 remaining runs (our DirichletLM baseline, USE_QA_T_C, and USE_QA_T_T). Additionally, we have 3 discussion features: (1) the length of a discussion, (2) the number of premises in a discussion, and (3) the mean length of premises of the discussion. While these discussion features are independent of the query, they can help build different rankers for different kinds of queries [21] (as determined by our query features) by dedicating different subtrees to different queries or discussions. Similarly, we have 3 query independent argument features: (1) the length of the premise, and (2) two features that specify the absolute, respectively the relative position of the argument in the discussion. Both discussion and argument features become only useful through our three

query features: (1) the number of words in a query, (2) the number of concepts, and (3) the number of entities (concepts and entities as determined with babelfy [22]). LambdaMART can dedicate different subtrees to different kinds of queries using the query features, e.g., for highly ambiguous queries (e.g., many entities), LambdaMART could learn that arguments from long discussions (many arguments) may be better suited than short discussions. As the last and lonely feature in our other category, we have the Boolean feature of whether an argument is in the garbage cluster as determined by our argument clustering or not.

We use the 31 features in a traditional learning-to-rank pipeline, reranking the top-100 results of our Multi-field Dirichlet-LM with LambdaMART. We train our model on the Task 1 relevance labels from Touché 2020 [14], discarding relevance labels that are not retrieved by our first-stage retrieval. The removal of relevance labels not occurring in our first-stage retrieval reduces the available training data from 2,298 labels to only 996 labels, but is necessary since we otherwise risk to introduce the selection bias found in the classical LETOR dataset [23] to our models (by showing the model many documents outside of the first-stage retrieval that it would never see during testing). We train all our LambdaMART models with RankLib [24] leaving all settings to their defaults.

Since we only have 996 labels to train and validate our model, we run a greedy feature selection algorithm on our 31 features to reduce the risk that our trained LambdaMART models overfit. In a 5 fold cross-validation setup (using the topics 1–10, 11–20, 21–30, 31–40, and 41–50 as folds), we start by selecting the feature with the highest nDCG@5 on our 996 labels. Afterwards, we select the next features by comparing the validation score of a LambdaMART model trained on the already selected feature(s) with remaining feature candidates, choosing always the feature with the highest mean nDCG@5 overall validation partitions of our 996 labels. During the execution of this greedy feature selection, we find that a set of 4 features (premise.dirichlet, is_garbage, conclusion.dfi, baseline) and a set of 9 features (the aforementioned 4 plus conclusion.bm25, premise.bm25, discussion_length, topic_string.dfi, num_premises) obtain the best validation scores in our experiments. For these two feature sets, we train two LambdaMART models without cross-validation using the first 40 topics for training and the remaining 10 topics for validation, yielding to our two LambdaMART runs: (1) lambdamart_small and (2) lambdamart_medium.

2.3. Reranking: Universal Sentence Encoder for Question-Answer Retrieval

This approach assumes argument retrieval to be sufficiently similar to question-answering (QA) to warrant the instrumentalization of off-the-shelf QA technology to retrieve arguments. We specifically use a retrieval question-answering variant of the Universal Sentence Encoder [12] used in Section 2.1. That model[5] consists of two separate encoders, one meant to encode a question and the other meant to encode a candidate response. Both question and candidate response are then projected into a 512-dimensional space where Euclidean distance is inversely proportional to how well the candidate response answers the question. We use the result set from Section 2.1 to embed every retrieved document using the response embedder. We then encode every topic query with the question embedder and compute the dot product of every document with every topic that retrieved it, then re-rank the arguments using this score.

[5] https://tfhub.dev/google/universal-sentence-encoder-qa/3

2.4. Query Expansion: Pattern-based RoBERTa Keyword-Generation

This approach is a refined version of the method we employed in the previous iteration of the workshop, as described in Section 3.2 in [25]. This approach employs a pattern (in that sense introduced by Schick and Schütze [13]) that embeds the topic into an argumentative dialectical context. We employ the list of patterns listed in Table 2. For every masked token of every template, we retrieve the top-20 most likely tokens returned by the masked-language-model head of RoBERTa [26]. This leaves us with 26 such top-20 lists of keywords which we combine into a single count dictionary indexed by keyword. We then concatenate all keywords into a single string, boosting[6] each one by its count from the dictionary, concatenate that result with the original topic query to form a new query which is then submitted to the retrieval system from Section 2.1. Consider the example topic *"Should professors get tenure?"* and a truncated list of the keyword dictionary this method generates:

> *education: 7, tenure: 7, accountability: 4, students: 4, teaching: 4, discipline: 4, children: 4, safety: 3, time: 3, security: 3, parents: 3, teachers: 3, benefits: 2, diversity: 2, retention: 2, experience: 2, compensation: 2, research: 2, term: 2, duration: 2, stigma: 2, retirement: 2, teacher: 2, math: 2, longevity: 1, flexibility: 1*

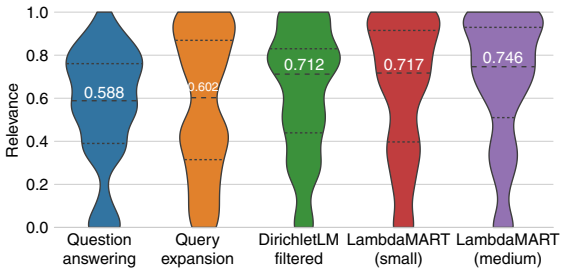
Table 2
Masked language model templates for query expansion. The masks (`<mask>`) indicate where the language model is tasked with guessing the most likely word out of its vocabulary.

Stance	Template
Positive	- **<query>** - Yes, because of **<mask>** and the benefits of **<mask> <mask>**
	- **<query>** - Absolutely, I think **<mask> <mask>** is **<mask>**
	- **<query>** - Yes, **<mask>** is associated with **<mask>** during **<mask>**.
Negative	- **<query>** - No, because of **<mask>** and the risk of **<mask> <mask>**.
	- **<query>** - Absolutely not, I think **<mask>** is **<mask> <mask>**
	- **<query>** - No, **<mask>** is associated with **<mask>** during **<mask>**.
Neutral	- **<query>** - What about **<mask>** or **<mask> <mask>**
	- **<query>** - Don't forget about **<mask>** and **<mask> <mask>**
	- **<query>** - This brings **<mask>** and **<mask>** to mind

[6] https://www.elastic.co/guide/en/elasticsearch/reference/7.4/query-dsl-query-string-query.html#_boosting

Table 3
Overview of the retrieval effectiveness in terms of relevance measured with nDCG@5. On the left, we report the mean nDCG@5 and confidence intervals of our 5 runs and two baselines and the winning run. On the right, we show violin plots for our 5 submitted runs with the median nDCG@5 highlighted.

Run	nDCG@5	$CI_{95\%}$
Best team (Elrond)	0.720	[0.654, 0.782]
LambdaMART (small)	0.678	[0.607, 0.746]
LambdaMART (medium)	0.647	[0.573, 0.714]
DirichletLM filtered	0.626	[0.557, 0.696]
Baseline DirichletLM	0.626	[0.550, 0.691]
Baseline args.me	0.607	[0.536, 0.679]
Query expansion	0.577	[0.497, 0.646]
Question answering	0.557	[0.498, 0.615]



3. Evaluation

We evaluate our five submitted runs and two official baselines using the official human judgments provided by the Touché organizers [3]. The official judgments provide quality and relevance labels for arguments using a top-5 pooling strategy. Hence, we follow the official evaluation and report the normalized discounted cumulative gain at a depth of 5 (nDCG@5) for the quality and relevance dimension. We include the two official baselines (one using a Dirichlet-based probabilistic similarity and the other using the args.me search engine API)[7] into our evaluations.

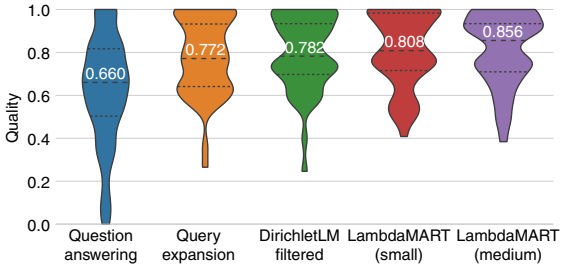
Table 3 provides an overview of how well each of our runs performed in terms of relevance measured at nDCG@5. Both LambdaMART runs improve upon the baseline DirichletLM which re-ranks the top-100 results. We observe that the LambdaMART (small) model with only 4 features is our best run in terms of relevance, obtaining substantially better nDCG@5 scores than baseline DirichletLM, and the LambdaMART (medium) model with 9 features. The substantial improvement in nDCG@5 (0.031 in the mean and 0.029 in the median) of the LambdaMART (small) model over the LambdaMART (medium) model indicates that 9 features already yield some overfitting in our cross-validation setup (which is plausible given only a few relevance labels were available). Still, we find that the DirichletLM implementation of the best team (obtaining an nDCG@5 of 0.720) substantially outperforms our LambdaMART models, indicating that improving the LambdaMART models using better features and more training data (which become available next year) is still possible. On the other side of the spectrum are template-based query expansion and re-ranking based on question answering, both of which exhibit worse effectiveness—respectively 0.577 and 0.557—in relation to the baseline which they both rely on. A satisfactory explanation lies in the unsupervised nature of both approaches and the absence of any fine-tuning on the argumentation domain. We expect both these approaches to improve by using human relevance judgments as a training signal.

Table 4 provides an overview of how well each of our runs performed in terms of quality measured at nDCG@5. We observe a correlation between quality and relevance: the ranking of our solutions along either dimension only differs in that the LambdaMART models exchange places. The LambdaMART (medium) model retrieves higher-quality arguments than its counterpart with 4 features—respectively 0.810 and 0.804 (mean nDCG@5). This correlation seems to imply a relation between the quality of an argument and its relevance to a particular topic.

[7] https://www.args.me/api-en.html

Table 4
Overview of the retrieval effectiveness in terms of quality measured with nDCG@5. On the left, we report the mean nDCG@5 and confidence intervals of our 5 runs and two baselines and the winning run. On the right, we show violin plots for our 5 submitted runs with the median nDCG@5 highlighted.

Run	nDCG@5	$CI_{95\%}$
Best team (Heimdall)	0.841	[0.803, 0.876]
LambdaMART (medium)	0.810	[0.771, 0.847]
LambdaMART (small)	0.804	[0.765, 0.842]
Baseline DirichletLM	0.796	[0.755, 0.838]
DirichletLM filtered	0.796	[0.756, 0.833]
Query expansion	0.779	[0.735, 0.817]
Baseline args.me	0.717	[0.657, 0.768]
Question answering	0.624	[0.566, 0.685]



4. Summary

Our participation in the first subtask of Touché 2021 consisted of five runs spanning four categories. First, we rely on the robustness of a multi-field retrieval DirichletLM index to return arguments from the args.me corpus, which we additionally sanitize using embeddings from Google's Universal Sentence Encoder, clustered using k-means. Second, we submit two runs generated with the feature-based learning to rank model LambdaMART, leveraging 4 respectively 9 features trained on Touché Topics 1 through 50 published in 2020. Third, we re-rank the results of the first method, using a score returned by a question-answering expert model, with the assumption that argument retrieval is close enough to the modality of that model to warrant such a use. Fourth, we leverage RoBERTa's masked language modeling ability to generate a weighted list of topical keywords that can be used to augment the original query. Altogether, the LambaMART models outperform the baseline, yet there is still room for improvement compared to the effectiveness of the respective best-performing approaches submitted.

References

[1] G. Blank, B. C. Reisdorf, The participatory web: A user perspective on web 2.0, Information, Communication & Society 15 (2012) 537–554.

[2] Y. Ajjour, H. Wachsmuth, J. Kiesel, M. Potthast, M. Hagen, B. Stein, Data Acquisition for Argument Search: The args.me corpus, in: C. Benzmüller, H. Stuckenschmidt (Eds.), 42nd German Conference on Artificial Intelligence (KI 2019), Springer, Berlin Heidelberg New York, 2019, pp. 48–59. doi:10.1007/978-3-030-30179-8_4.

[3] A. Bondarenko, L. Gienapp, M. Fröbe, M. Beloucif, Y. Ajjour, A. Panchenko, C. Biemann, B. Stein, H. Wachsmuth, M. Potthast, M. Hagen, Overview of Touché 2021: Argument Retrieval, in: D. Hiemstra, M.-F. Moens, J. Mothe, R. Perego, M. Potthast, F. Sebastiani (Eds.), Advances in Information Retrieval. 43rd European Conference on IR Research (ECIR 2021), volume 12036 of *Lecture Notes in Computer Science*, Springer, Berlin Heidelberg New York, 2021, pp. 574–582. URL: https://link.springer.com/chapter/10.1007/978-3-030-72240-1_67. doi:10.1007/978-3-030-72240-1_67.

[4] M. Potthast, L. Gienapp, F. Euchner, N. Heilenkötter, N. Weidmann, H. Wachsmuth,

B. Stein, M. Hagen, Argument search: Assessing argument relevance, in: B. Piwowarski, M. Chevalier, É. Gaussier, Y. Maarek, J. Nie, F. Scholer (Eds.), Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2019, Paris, France, July 21-25, 2019, ACM, 2019, pp. 1117–1120. URL: https://doi.org/10.1145/3331184.3331327. doi:10.1145/3331184.3331327.

[5] M. Potthast, M. Hagen, B. Stein, The Dilemma of the Direct Answer, SIGIR Forum 54 (2020). URL: http://sigir.org/forum/issues/june-2020/.

[6] H. Wachsmuth, N. Naderi, Y. Hou, Y. Bilu, V. Prabhakaran, T. A. Thijm, G. Hirst, B. Stein, Computational Argumentation Quality Assessment in Natural Language, in: P. Blunsom, A. Koller, M. Lapata (Eds.), 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2017), 2017, pp. 176–187. URL: http://aclweb.org/anthology/E17-1017.

[7] H. Wachsmuth, M. Potthast, K. Al-Khatib, Y. Ajjour, J. Puschmann, J. Qu, J. Dorsch, V. Morari, J. Bevendorff, B. Stein, Building an Argument Search Engine for the Web, in: K. Ashley, C. Cardie, N. Green, I. Gurevych, I. Habernal, D. Litman, G. Petasis, C. Reed, N. Slonim, V. Walker (Eds.), 4th Workshop on Argument Mining (ArgMining 2017) at EMNLP, Association for Computational Linguistics, 2017, pp. 49–59. URL: https://www.aclweb.org/anthology/W17-5106.

[8] L. Ein-Dor, E. Shnarch, L. Dankin, A. Halfon, B. Sznajder, A. Gera, C. Alzate, M. Gleize, L. Choshen, Y. Hou, Y. Bilu, R. Aharonov, N. Slonim, Corpus wide argument mining - A working solution, in: The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020, AAAI Press, 2020, pp. 7683–7691. URL: https://aaai.org/ojs/index.php/AAAI/article/view/6270.

[9] C. Zhai, J. Lafferty, A study of smoothing methods for language models applied to ad hoc information retrieval, in: Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '01, Association for Computing Machinery, New York, NY, USA, 2001, pp. 334–342. URL: https://doi.org/10.1145/383952.384019. doi:10.1145/383952.384019.

[10] C. J. C. Burges, K. M. Svore, Q. Wu, J. Gao, Ranking, Boosting, and Model Adaptation, Technical Report MSR-TR-2008-109, 2008. URL: https://www.microsoft.com/en-us/research/publication/ranking-boosting-and-model-adaptation/.

[11] D. Cer, Y. Yang, S. Kong, N. Hua, N. Limtiaco, R. S. John, N. Constant, M. Guajardo-Cespedes, S. Yuan, C. Tar, Y. Sung, B. Strope, R. Kurzweil, Universal sentence encoder, CoRR abs/1803.11175 (2018). URL: http://arxiv.org/abs/1803.11175. arXiv:1803.11175.

[12] Y. Yang, D. Cer, A. Ahmad, M. Guo, J. Law, N. Constant, G. Hernandez Abrego, S. Yuan, C. Tar, Y.-h. Sung, B. Strope, R. Kurzweil, Multilingual universal sentence encoder for semantic retrieval, in: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations, Association for Computational Linguistics, Online, 2020, pp. 87–94. URL: https://www.aclweb.org/anthology/2020.acl-demos.12. doi:10.18653/v1/2020.acl-demos.12.

[13] T. Schick, H. Schütze, Exploiting cloze-questions for few-shot text classification and natural language inference, in: Proceedings of the 16th Conference of the European

Chapter of the Association for Computational Linguistics: Main Volume, Association for Computational Linguistics, Online, 2021, pp. 255–269. URL: https://www.aclweb.org/anthology/2021.eacl-main.20.

[14] A. Bondarenko, M. Fröbe, M. Beloucif, L. Gienapp, Y. Ajjour, A. Panchenko, C. Biemann, B. Stein, H. Wachsmuth, M. Potthast, M. Hagen, Overview of Touché 2020: Argument Retrieval, in: Working Notes Papers of the CLEF 2020 Evaluation Labs, 2020.

[15] J. Devlin, M. Chang, K. Lee, K. Toutanova, BERT: pre-training of deep bidirectional transformers for language understanding, CoRR abs/1810.04805 (2018). URL: http://arxiv.org/abs/1810.04805. arXiv:1810.04805.

[16] R. Kiros, Y. Zhu, R. Salakhutdinov, R. S. Zemel, R. Urtasun, A. Torralba, S. Fidler, Skip-thought vectors, in: C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, R. Garnett (Eds.), Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada, 2015, pp. 3294–3302. URL: http://papers.nips.cc/paper/5950-skip-thought-vectors.

[17] M. L. Henderson, R. Al-Rfou, B. Strope, Y. Sung, L. Lukács, R. Guo, S. Kumar, B. Miklos, R. Kurzweil, Efficient natural language response suggestion for smart reply, CoRR abs/1705.00652 (2017). URL: http://arxiv.org/abs/1705.00652. arXiv:1705.00652.

[18] S. R. Bowman, G. Angeli, C. Potts, C. D. Manning, A large annotated corpus for learning natural language inference, in: L. Màrquez, C. Callison-Burch, J. Su, D. Pighin, Y. Marton (Eds.), Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015, The Association for Computational Linguistics, 2015, pp. 632–642. URL: https://doi.org/10.18653/v1/d15-1075. doi:10.18653/v1/d15-1075.

[19] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: Machine learning in Python, Journal of Machine Learning Research 12 (2011) 2825–2830.

[20] S. Clinchant, É. Gaussier, Information-based models for ad hoc IR, in: F. Crestani, S. Marchand-Maillet, H. Chen, E. N. Efthimiadis, J. Savoy (Eds.), Proceeding of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2010, Geneva, Switzerland, July 19-23, 2010, ACM, 2010, pp. 234–241. URL: https://doi.org/10.1145/1835449.1835490. doi:10.1145/1835449.1835490.

[21] C. Macdonald, R. L. T. Santos, I. Ounis, On the usefulness of query features for learning to rank, in: X. Chen, G. Lebanon, H. Wang, M. J. Zaki (Eds.), 21st ACM International Conference on Information and Knowledge Management, CIKM'12, Maui, HI, USA, October 29 - November 02, 2012, ACM, 2012, pp. 2559–2562. URL: https://doi.org/10.1145/2396761.2398691. doi:10.1145/2396761.2398691.

[22] A. Moro, A. Raganato, R. Navigli, Entity linking meets word sense disambiguation: a unified approach, Trans. Assoc. Comput. Linguistics 2 (2014) 231–244. URL: https://tacl2013.cs.columbia.edu/ojs/index.php/tacl/article/view/291.

[23] T. Minka, S. Robertson, Selection bias in the letor datasets, in: SIGIR Workshop on Learning to Rank for Information Retrieval, ACM, Singapore, 2008, pp. 48–51.

[24] V. Dang, The lemur project-wiki-ranklib, Lemur Project (2013). Available: https://sourceforge.net/p/lemur/wiki/RankLib.

[25] C. Akiki, M. Potthast, Exploring Argument Retrieval with Transformers, in: L. Cappellato, C. Eickhoff, N. Ferro, A. Névéol (Eds.), Working Notes Papers of the CLEF 2020 Evaluation Labs, volume 2696, 2020. URL: http://ceur-ws.org/Vol-2696/.

[26] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, V. Stoyanov, Roberta: A robustly optimized BERT pretraining approach, CoRR abs/1907.11692 (2019). URL: http://arxiv.org/abs/1907.11692. arXiv:1907.11692.