Proceedings of the 12th Conference on Language Resources and Evaluation (LREC 2020), pages 1926–1933
Marseille, 11–16 May 2020

WIKIR: A Python toolkit for building a large-scale Wikipedia-based English Information Retrieval Dataset

Jibril Frej, Didier Schwab, Jean-Pierre Chevallet
Univ. Grenoble Alpes, CNRS, Grenoble INP*, LIG, 38000 Grenoble, France
* Institute of Engineering Univ. Grenoble Alpes
{jibril.frej, didier.schwab, jean-pierre.chevallet}@univ-grenoble-alpes.fr

Abstract

Over the past years, deep learning methods allowed for new state-of-the-art results in *ad-hoc* information retrieval. However such methods usually require large amounts of annotated data to be effective. Since most standard *ad-hoc* information retrieval datasets publicly available for academic research (e.g. *Robust04*, *ClueWeb09*) have at most 250 annotated queries, the recent deep learning models for information retrieval perform poorly on these datasets. These models (e.g. DUET, Conv-KNRM) are trained and evaluated on data collected from commercial search engines not publicly available for academic research which is a problem for reproducibility and the advancement of research. In this paper, we propose *WIKIR*: an open-source toolkit to automatically build large-scale English information retrieval datasets based on *Wikipedia*. *WIKIR* is publicly available on *GitHub*. We also provide wikIR78k and wikIRS78k: two large-scale publicly available datasets that both contain 78,628 queries and 3,060,191 (query, relevant documents) pairs.
Keywords: Information Retrieval, Open Source, Dataset, Deep Learning

1. Introduction

Deep learning has been shown to be effective in various natural language processing (NLP) tasks such as language modeling, reading comprehension, question answering and natural language understanding (Devlin et al., 2019; Yang et al., 2019b). However, both large and public datasets are key factors for developing effective and reproducible deep learning models.

Ad-hoc information retrieval (IR) consists in ranking a set of unstructured documents with respect to a query. Despite the progress in NLP using deep neural networks (DNNs), *ad-hoc* IR on text documents has not benefited as much as other fields of NLP from DNNs yet (Dehghani et al., 2017). The absence of significant success in *ad-hoc* IR using deep learning approaches is mainly due to the complexity of solving the ranking task using only unlabelled data (Dehghani et al., 2017). Consequently, the availability of large amount of labelled data is crucial to develop effective DNNs for *ad-hoc* IR. However, as described in Table 1, most of the publicly available English IR datasets only have few labelled data with at most 1,692 labelled queries.

Other datasets than the ones presented in Table 1, such as *Yahoo! LETOR* (Chapelle and Chang, 2011), with more labelled data (≈30k labelled queries) are publicly available. However, only the feature vectors describing query-document pairs are provided. Such datasets are suitable for feature-based learning-to-rank models but not for DNNs that require the original content of queries and documents.

Thus, most of the deep learning model for *ad-hoc* IR that have been proposed recently are developed using one of the following approaches:

(1) Using large amounts of data collected from commercial search engines that are not publicly available (Yang et al., 2019a; Mitra et al., 2017). This process is expensive, time consuming and not reproducible.

(2) Using publicly available datasets that have few annotated data such as *MQ2007* and *MQ2008* (Pang et al., 2017; Fan et al., 2018). This approach can restrain the model design due to the lack of data.

(3) Using weak supervision that consists in pre-training a supervised model on data labelled with an unsupervised approach (Dehghani et al., 2017). However, this method can bias large models to rank similarly as the unsupervised ranker.

Recently, Zheng et al. (2018) proposed *Sogou-QCL*, a publicly available dataset in Chinese with click relevance label. To the best of our knowledge, *Sogou-QCL* is the only public large-scale (≈500k queries) dataset for *ad-hoc* IR. The release of this dataset was the first step in reproducible research on neural ranking model applied to *ad-hoc* IR.

Wikipedia has recently been used to build large-scale cross-lingual information retrieval (CLIR) datasets to train effective neural learning-to-rank models (Schamoni et al., 2014). Leveraging this idea, we propose *WIKIR*: a toolkit to build a *Wikipedia*-based large-scale English IR dataset. *WIKIR* can also be used to train and evaluate several deep text matching models on the datasets it created.

Moreover, we propose a general framework to build IR datasets automatically from any set of documents constrained by three topical properties that will be introduced further (see Section 2.1.).

Our contributions are fourfold:

- We provide *WIKIR*: a toolkit[1] to build a *Wikipedia*-based English Information Retrieval dataset;
- We present a framework for creating IR datasets from a set of documents that satisfies three topical properties: *Existence*, *Identifiability* and *Describability*;
- We propose wikIR78k and wikIRS78k: two large-scale datasets generated with *WIKIR*, publicly available for download[2,3];

[1] https://github.com/getalp/wikIR
[2] https://www.zenodo.org/record/3707606
[3] https://www.zenodo.org/record/3707238

Dataset	#Query	#Doc	Avg $\#d^+/q$
CLEF 2014	50	1M	64.56
ClueWeb09	200	1B	74.62
ClueWeb12	100	733M	189.63
GOV2	150	25M	181.51
MQ2007	1,692	65k	10.63
MQ2008	784	14k	3.82
Robust04	250	0.5M	63.28

Table 1: Statistics of several publicly available English IR Dataset where the original query and document contents are available. Avg $\#d^+/q$ denotes the average number of relevant document per query.

- We provide *Python* scripts to train and evaluate deep learning models for *ad-hoc* IR on our datasets.

2. A general framework for automatic IR dataset creation

In this section, we propose a general framework to create automatically an IR dataset from a resource $\mathcal{R}$ composed of a set of documents. An IR dataset is composed of:

- $\mathcal{D}$, a set of documents;
- $\mathcal{Q}$, a set of queries;
- $\mathcal{R}el$, a set of relevance labels for each query-document pairs (Schütze et al., 2008).

2.1. Properties

We define 3 properties that $\mathcal{R}$ must satisfy to be used to build an IR dataset.

Topical Existence. There exists at least one topic related to each document in $\mathcal{R}$.
Topical Existence guarantees the topical relevance (Mizzaro, 1997) of documents with respect to a subject.

Topical Identifiability. There exists a function $identify()$ that identifies all the topics related to any document of $\mathcal{R}$. Using *Topical Identifiability*, we can assess the relevance of documents with respect to the topics in $\mathcal{R}$.

Topical Describability. There exists a function $describe()$ that associates every topic with a short and accurate description.
Topical Describability is desirable to be able to build queries from the topics in the resource $\mathcal{R}$.

2.2. Dataset construction

In the following, we describe how to use a resource $\mathcal{R}$ that satisfies the three properties listed above to automatically construct an IR dataset.

Document construction. We choose a subset of the resource $\mathcal{R}$ to construct the set of documents: $\mathcal{D} \subseteq \mathcal{R}$. For example, if $\mathcal{R}$ is the set of *Wikipedia* articles, we can choose $\mathcal{D}$ to be the set of *Wikipedia* articles that contain more than 1000 words.

Query construction. We start by identifying all topics in the set of documents $\mathcal{D}$ using the $identify()$ function:

$$\mathcal{T}_\mathcal{D} = \bigcup_{d \in \mathcal{D}} identify(d),$$

where $\mathcal{T}_\mathcal{D}$ is the set of all topics in $\mathcal{D}$. Then, we use the $describe()$ function on all of the topic to construct the query set $\mathcal{Q}$:

$$\mathcal{Q} = \left\{ describe(t) \middle| t \in \mathcal{T}_\mathcal{D} \right\}.$$

Relevance label construction. $\mathcal{R}el$ is the set of all (query-document-relevance) triplets:

$$\mathcal{R}el = \left\{ \big(q, d, rel(q,d)\big) \middle| \big(q,d\big) \in \mathcal{Q} \times \mathcal{D} \right\},$$

with $rel()$ a function that associates every query-document pairs with a relevance label. We propose to assign a positive relevance label (denoted val^+) to document d with respect to query q if d contains the topic that was used to build q. Otherwise a negative or null relevance label (denoted val^-) is assigned:

$$rel(q,d) = \begin{cases} val^+ \in \mathbb{R}^+_*, & \text{if } t_q \in identify(d), \\ val^- \in \mathbb{R}^-, & \text{else,} \end{cases}$$

where t_q stands for the topic that was used to build query q: $describe(t_q) = q$

2.3. The case of *Wikipedia*

In this subsection we show that the set of English *Wikipedia* articles W does satisfy *Topical Existence*, *Describability* and *Identifiability*. A simplified description of the construction process of an IR dataset using 2 articles from *Wikipedia* is displayed in Figure 1.
Topical Existence. Every *Wikipedia* article is related to at least one topic: its main subject.
Topical Identifiability. We assume that if an article a contains an internal link to another article a_t in its first sentence (denoted f_a), then the main subject of a_t is a topic of a. The intuition behind this assumption is that the first sentence of most *Wikipedia* articles is a good description of the article's content (Sasaki et al., 2018) and if a link is present, it points to an important topic of the considered article. Therefore, we propose to define $identify()$ as follows:

$$identify(a) = \{s_a\} \bigcup \left\{ s_{a_t} \in \mathcal{W} \middle| \exists\ f_{a_t} \xrightarrow[\text{link}]{} a \right\}, \quad (1)$$

where s_a denotes the main subject of article a and $f_{a_t} \xrightarrow[\text{link}]{} a$ designs an internal link in the first sentence of article a_t that points to article a. Thus, $identify()$ considers the set of topics related to article a as the main subject of a: s_a and the main subject of all articles that points to a in their first sentence. For example, the set of topics related to the article *Developmental disorder* is its main subject and the main subject of the article *Autism* because there is a link in the first sentence of article *Autism* that points to article *Developmental disorder* (see Figure 1).

Topical Describability. Because topics are main subjects of *Wikipedia* articles, one way to get a short and accurate description is to use the article title:

$$describe(s_a) = title_a, \quad (2)$$

where $title_a$ is the title of article a. To get a long and noisy topic description, we can also use the article first sentence:

$$describe(s_a) = f_{a_t}. \quad (3)$$

3. WIKIR toolkit description

In this section, we describe *WIKIR* toolkit and make explicit the motivations behind some design decisions. For an exhaustive list of the options available and to have examples on how to use *WIKIR*, please check our *github* repository: https://github.com/getalp/wikIR

3.1. WIKIR for dataset creation

To create a dataset using an XML *Wikipedia* dump file from *Wikimedia* database backup dumps.[4] *WIKIR* follows 3 main steps: construction, processing and storing.

3.1.1. Dataset construction

Wikipedia dump extraction. We use *WikiExtractor*[5] to extract plain text from an English *Wikipedia* dump. We end up with a *json* file (described in Figure 2) that contains the URL, title and text of all *Wikipedia* articles. When using *wikiextractor*, we use the option to preserve links in the text in order to build relevance labels.

Document extraction. The set of documents $\mathcal{D}$ is extracted using the "text" field associated to each article in the *json* file produced by the previous step. The first line of the "text" field (that corresponds to the article title) is deleted. We also remove article title from documents in order to avoid the following situation: given a query, the most relevant document will always starts with the query itself which makes the ranking task significantly easier.

Query construction. As described in Section 2.2. to build queries we need an $identify()$ function and a $describe()$ function. *WIKIR* uses the $identify()$ function defined in equation (1). The $describe()$ function is defined using equation (2) or equation (3). To sum up, topics are identified using internal links and are described using either article titles or article first sentences. The construction process of queries is the same as in Section 2.2.

Relevance label construction. As explained in Section 2.2. in order to build $\mathcal{R}el$ we need to define $rel()$. To do so, we assume that the most relevant document for a query is the document built from the same article as the query. Consequently, we define $rel()$ as:

$$rel(q,d) = \begin{cases} 2, & \text{if } a_q = a_d, \\ 1, & \text{if } a_d \in identify(d) \setminus a_d, \\ 0, & \text{otherwise,} \end{cases}$$

where a_q (resp. a_d) denotes the *Wikipedia* article used to build query q (resp. document d). Thus we assign a relevance label equal to two for query-document pairs that come from the same article. We assign a relevance label equal to one to a query-document pair if there is a link from the first sentence of the article of the document that points to the article of the query. For example, if we consider the query *"Developmental disorder"*, the most relevant (relevance = 2) document is *"Developmental disorders comprise a group of . . . "* because they are built from the same the article. The document *"Autism is a developmental disorder characterized by . . . "* is relevant (relevance = 1) because the article *Autism* contains a link to the *Developmental disorder* article (see Figure 1).

3.1.2. Dataset processing

Query selection. In order to have a balanced dataset, we select only queries that have a minimum number of relevant documents (5 by default). We also limited queries length to a maximum of 10 words.

Preprocessing. *WIKIR* starts by deleting the target in hypertext references (*href*) but keeps the text. For example, *"worsening"* becomes *"worsening"*. Then, every non alphanumerical character is deleted. By default *WIKIR* also lowercases all the characters in the dataset.

Separation into training, validation and test sets. Queries and their corresponding relevance label (*qrels*) are randomly separated into training, validation and test sets. Documents are not separated as well because in *ad-hoc* IR, we assume to have a fixed set of documents to retrieve from (Baeza-Yates and Ribeiro-Neto, 1999).

3.2. WIKIR for BM25: a first stage ranker

3.2.1. Motivation

After the dataset is created, *WIKIR* can be used to run Okapi BM25 (Robertson and Walker, 1994): a state-of-the art IR model compatible with an inverted index. An inverted index is a structure to store the documents of an IR dataset that makes the retrieval of documents extremely efficient (Sanderson, 2010). We propose this option because the vast majority of DNNs developed for *ad-hoc* IR are not compatible with an inverted index (Zamani et al., 2018). They rely on a first ranking stage made by an efficient model such as BM25 and only re-rank the top-k documents for a given query in order to have an efficient search. Thus *WIKIR* can be used to run BM25 and save the top-k documents for each query.

3.2.2. Implementation

Instead of using a common information retrieval system (IRS) such as *Terrier*,[6] *Lucene*[7] or *Lemur*[8] to run and evaluate BM25 on our dataset, we used the *Python* library *Rank-BM25*.[9] We made this decision to facilitate the use of *WIKIR* and to aid the reproducibility of our experiments that do not require the installation of any software that is

[4]https://dumps.wikimedia.org/backup-index.html
[5]https://github.com/attardi/wikiextractor

[6]http://terrier.org/
[7]http://lucene.apache.org/
[8]http://www.lemurproject.org
[9]https://github.com/dorianbrown/rank_bm25

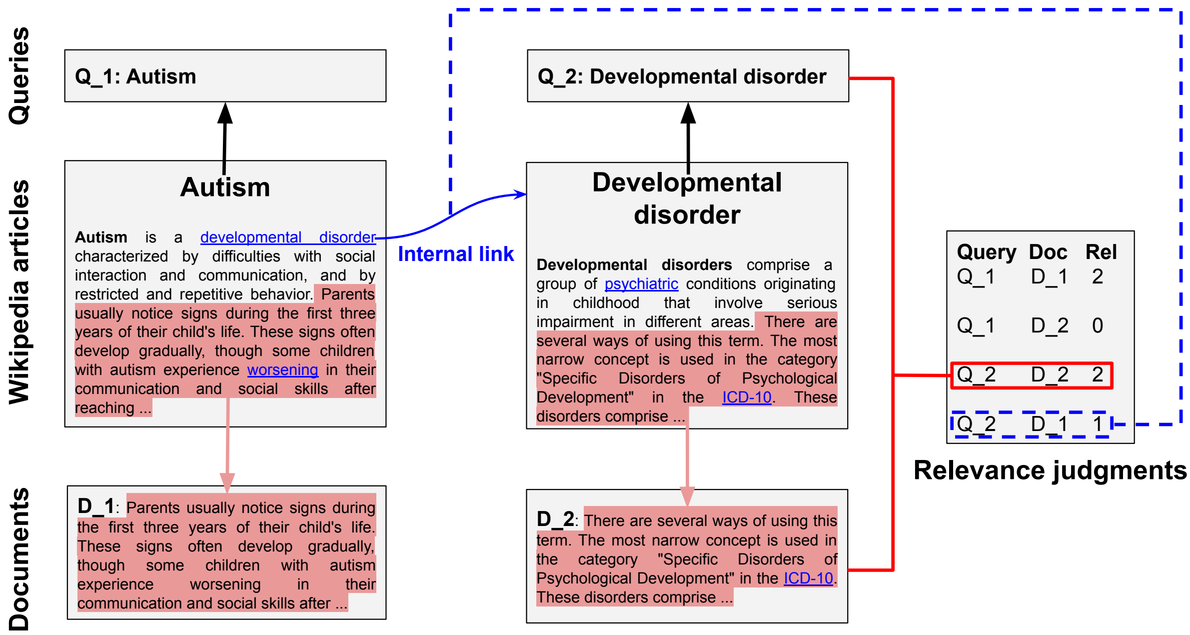

Figure 1: Description of the construction process of an IR dataset by *WIKIR* using only two articles. Queries are built from the title of articles. Documents are constructed using the full text of articles without the title and without the first sentence. A relevance label equal to 2 is assigned to query and documents that are built from the same article. A relevance label equal to 1 is assigned using internal links in the first sentence of articles.

```
{"id": "12",
 "url": "https://en.wikipedia.org/wiki?curid=12",
 "title": "Anarchism",
 "text": "Anarchism\n\nAnarchism is an <a href=\"anti-authoritarian\">anti-
    authoritarian</a> <a href=\"political%20philosophy\">political philosophy
    </a> that advocates ... "
}
{"id": "25",
 "url": "https://en.wikipedia.org/wiki?curid=25",
 "title": "Autism",
 "text": ""Autism\n\nAutism is a <a href=\"developmental%20disorder\">
    developmental disorder</a> characterized by difficulties with ... "
}
...
```

Figure 2: *json* file extracted from English *Wikipedia* dump using *WikiExtractor*

not in our *GitHub* repository. Because *Rank-BM25* does not preprocess text, we used *nltk Python* library (Loper and Bird, 2002) to apply Porter stemmer (Porter, 2001) and stopword removal as commonly done in IR. It should be noted that we applied stemming and stopword removal only for BM25: the queries and documents in the dataset created by *WIKIR* are not stemmed and do contain stopwords.

3.3. WIKIR for neural re-ranking

WIKIR can be used to train and evaluate DNNs on the dataset it created. As explained in Section 3.2., we perform neural re-ranking using BM25 as a first stage ranker. We used *MatchZoo* deep text matching library for training and evaluation of the models. We used *MatchZoo* because it has been accepted as a reliable toolkit for deep text matching research (Guo et al., 2019b). Any model available in *MatchZoo* can be trained and evaluated with *WIKIR*. Once the training is done and the rankings of documents are saved, our toolkit can be used to compute evaluation measures, statistical significance and display the performance of each model in a format compatible with a LaTeX table.

4. Datasets

In this section, we describe wikIR78k and wikIRS78k: the two datasets created by *WIKIR* that we used in our experiments.

	wikIRS78k	wikIR78k
Document count	2.4M	2.4M
Average document length	744.58	744.58
Query count	78k	78k
Average query length	2.45	9.80
Avg #d^+/q	39.02	39.02

Table 2: Statistics of wikIR78k and wikIRS78k. Avg #d^+/q denotes the average number of relevant document per query.

wikIR78k. wikIR78k is a large-scale dataset that contains 78,631 annotated queries. To build wikIR78k, we used the full set of *Wikipedia* articles. To build queries, we used article titles. Moreover, we deleted the first sentence of each article when constructing the documents. We made this choice since all the information we use to assess relevance is contained in the first sentence of articles (see Section 2.2.) and we do not want DNNs that take into account word order to use this bias to their advantage.
wikIRS78k. The construction process of wikIRS78k is the same as wikIR78k, with the exception of queries construction: we used articles first sentences instead of article titles. We propose a dataset with short and well defined queries and a dataset with long and noisy queries to study the robustness of IR models against noisy queries. Statistics of the datasets are displayed on Table 2. Queries are randomly split into training, validation and tests sets of size 80% ,10% ,10% respectively.

5. Experimental settings

This section describes the experiments we conducted on our datasets.

5.1. Models description

We evaluated 3 types of models: bag-of-words, DNNs for text matching and DNNs for *ad-hoc* IR.

5.1.1. Exact matching model

We use Okapi BM25: a state-of-the-art ranking function that uses exact matches between query and document terms (Robertson and Walker, 1994):

$$\text{BM25}(q,d) = \sum_{t \in q} \text{idf}_t \frac{\text{tf}_{td}(k_1+1)}{\text{tf}_{td} + k_1\left(1 - b + b\frac{|d|}{avgdl}\right)}, \quad (4)$$

where q is a query, d is a document, tf_{td} is the term frequency (number of occurrences) of term t in document d, k_1 and b are hyperparameters of BM25 and $avgdl$ denotes the average length of documents in C. The inverse document frequency of term t denoted as idf_t reflects the discriminative power of term t to assess relevance (Schütze et al., 2008):

$$\text{idf}_t = \log \frac{|C|+1}{\text{df}_t}, \quad (5)$$

where C is the considered collection of documents and df_t is the document frequency of term t: the number of documents that contain term t.

5.1.2. Deep neural networks for text matching

Text matching is a general task that consists in computing a matching score between two texts. Models developed for text matching do not take into account IR specificities such as query term importance or exact matching signals consideration (Guo et al., 2016).
ArcI. A representation model that uses 1D-convolutions and pooling layers to get a fixed size representation of sentences. The similarity score is obtained with a multilayer perceptron (MLP) on the representations of the two inputs (Hu et al., 2014).
ArcII. An interaction model that uses 1D-convolutions to build an interaction matrix of the two input sentences. The final score is obtained using 2D-convolutions, max-pooling and MLP on the interaction matrix (Hu et al., 2014).
MatchPyramid. An interaction model that build an interaction matrix between the two input sentences using the dot product between their word embeddings. The matrix obtained is processed using a convolutional neural network (CNN) and the matching score is computed using a MLP on the output of the CNN (Pang et al., 2016).

5.1.3. Deep neural networks for *ad-hoc* IR

DRMM. Uses a matching histogram between query term and all of the document terms, followed by a MLP to get a query term score. The final matching score is the sum of all query terms scores (Guo et al., 2016).
KNRM. A neural ranking model that uses word interactions and kernel pooling to produce learning-to-rank features. The final score is computed with a linear layer and a non-linear activation function applied on the ranking features (Xiong et al., 2017).
DUET. Model that uses both local (exact matching of n-grams of characters) and distributed (word embeddings) representations to compute a relevance score (Mitra et al., 2017).
Conv-KNRM. As KNRM, Conv-KNRM (Dai et al., 2018) is based on kernel pooling to produce learning-to-rank features but it uses convolutions to match n-grams of words and has multiple interaction matrices.

5.2. Implementation details

Training. Each training sample consists of a query q, a document d^+ relevant to q and a set of 5 irrelevant documents D^- with respect to q. We use the cross entropy loss function for ranking provided by *MatchZoo* defined as:

$$\mathcal{L}(q, d^+, D^-) = rel(q, d^+) \log \frac{\exp\left(s(q, d^+)\right)}{\sum\limits_{d^- \in D^-} \exp\left(s(q, d^-)\right)}$$

where $s(q, d)$ denoted the score of d with respect to q. We used the cross entropy loss function for ranking instead of the widely used Hinge loss function for pairwise training of *ad-hoc* IR models (Guo et al., 2019a) as preliminary experiments showed that the cross entropy loss function is more efficient in terms of training time and produces more effective models. We use the Adam optimizer (Kingma and Ba, 2015) with a learning rate equals to 0.001. Each model is trained 5 times (with different initialization) for 50 epochs.

wikIR78k

Model	P@5	P@10	P@20	nDCG@5	nDCG@10	nDCG@20	nDCG	MAP
BM25	0.2622	0.2039	0.1498	0.3269	0.3045	0.3098	0.3555	0.1498
ArcI	0.1412-	0.1316-	0.1171-	0.1393-	0.1510-	0.1749-	0.2537-	0.0841-
ArcII	0.1492-	0.1401-	0.1224-	0.1428-	0.1559-	0.1799-	0.2560-	0.0885-
MatchPyramid	0.2302-	0.1886-	0.1485	0.2568-	0.2495-	0.2644-	0.3160-	0.1253-
KNRM	0.1288-	0.1199-	0.1078-	0.1186-	0.1296-	0.1531-	0.2402-	0.0761-
DUET	0.2645	0.2038	0.1533+	0.3323	0.3044	0.3082	0.3533	0.1447-
DRMM	**0.2760+**	**0.2122+**	0.1548+	**0.3462+**	**0.3189+**	**0.3227+**	**0.3653+**	**0.1566+**
Conv-KNRM	0.2602	0.2057	**0.1566+**	0.3080-	0.2906-	0.2992-	0.3422-	0.1419-

wikIRS78k

Model	P@5	P@10	P@20	nDCG@5	nDCG@10	nDCG@20	nDCG	MAP
BM25	0.2177	0.1634	0.1186	0.2944	0.2673	0.2695	0.3085	0.1163
ArcI	0.1156-	0.1076-	0.0953-	0.1096-	0.1201-	0.1418-	0.2104-	0.0650-
ArcII	0.1360-	0.1236-	0.1055-	0.1299-	0.1397-	0.1602-	0.2210-	0.0726-
MatchPyramid	0.2053-	0.1665	0.1271+	0.2296-	0.2232-	0.2336-	0.2722-	0.1025-
KNRM	0.1443-	0.1239-	0.1010-	0.1501-	0.1541-	0.1705-	0.2315-	0.0758-
DUET	0.2534+	0.1926+	0.1387+	0.3252+	0.2964+	0.2951+	0.3207+	0.1294+
DRMM	0.2368+	0.1769+	0.1275+	0.3188+	0.2872+	0.2868+	0.3197+	0.1248+
Conv-KNRM	**0.2661+**	**0.2026+**	**0.1458+**	**0.3253+**	**0.3004+**	**0.3010+**	**0.3223+**	**0.1351+**

Table 3: Performance comparison of different models on wikIR78k and wikIRS78k. Significant improvement/degradation with respect to BM25 is denoted as (+/-) with p-value < 0.01.

We select the model that has the highest normalized discounted cumulative gain (Järvelin and Kekäläinen, 2002) on the validation set and report its results on the test set.

Embeddings. We used Glove (Pennington et al., 2014) word embeddings of dimension 300 provided by *MatchZoo*.

Hyperparameters. BM25 hyperparameters are set to their default values in *Rank-BM25*: $k_1 = 1.5$ and $b = 0.75$. Hyperparameters associated with DNNs (e.g., number of layers, kernel size, similarity function) were set to their default value implemented in *MatchZoo*, except for the dropout rate that we set to 0.5 for models with a dropout parameter.

Evaluation metrics. We use 3 standard evaluation metrics: MAP, Precision and normalized discounted cumulative gain (nDCG). We use a two-tailed paired t-test with Bonferroni correction to measure statistically significant differences between the evaluation metrics (Urbano et al., 2013; Fuhr, 2018).

6. Results and discussion

6.1. Short and well defined queries

As we can see on Table 3, when queries are short and well defined (wikIR78k) BM25 is a strong baseline. Indeed, only the DRMM model manages to outperform BM25 on all metrics with statistical significance. Moreover, even though the DUET and Conv-KNRM models were designed for *ad-hoc* IR, they do not manage to outperform BM25.

Models that were not designed for *ad-hoc* IR but for text matching perform statistically significantly worst than BM25. This suggests that datasets created with *WIKIR* are suited for designing and training DNNs specifically for *ad-hoc* IR.

6.2. Long and noisy queries

Interestingly, models react differently to noisy queries (wikIRS78k). BM25 and DRMM are strongly affected by noise (-9.94% and -7.91%, respectively on the nDCG@5 compared to wikIR78k) whereas KNRM and Conv-KNRM have better performances on noisy queries ($+26.56\%$ and $+5.32\%$, respectively on the nDCG@5 compared to wikIR78k). Moreover, with the exception of KNRM, all models designed specifically for *ad-hoc* IR perform better than BM25 on all metrics with statistical significance. However DRMM does not achieve the best performances anymore. This indicates that DRMM is best suited for short and well defined queries but other models with more parameters such as Conv-KNRM and DUET are more robust to noise given enough training data.

7. Conclusions and future work

In this paper, we propose *WIKIR* a toolkit for building large-scale English information retrieval dataset from *Wikipedia*. *WIKIR* can also be used to train and evaluate deep text matching models. We propose a general framework to construct an IR dataset from any resource that satisfies three topical properties. Additionally, we made available for download wikIR78k and wikIRS78k: two large-scale IR datasets built using *WIKIR*, that are well suited for designing and training deep models for *ad-hoc* IR. All our code is available and our experiments are reproducible.

For future work, we plan to use wikIR78k and wikIRS78k to pre-train deep models for *ad-hoc* IR and fine-tune them on standard IR datasets to see if any gain is obtained compared to weak supervision (Dehghani et al., 2017). We will also adapt *WIKIR* to more languages and try our framework to produce IR datasets from other resources such as PubMed Central.[10]

8. Acknowledgements

The authors would like to thank Maximin Coavoux,[11] Emmanuelle Esperança-Rodier,[11] Lorraine Goeuriot,[11] William N. Havard,[11] Quentin Legros,[12] Fabien Ringeval,[11] and Loïc Vial[11] for their thoughtful comments and efforts towards improving our manuscript.

9. Bibliographical References

Baeza-Yates, R. A. and Ribeiro-Neto, B. A. (1999). *Modern Information Retrieval*. ACM Press / Addison-Wesley.

Chapelle, O. and Chang, Y. (2011). Yahoo! learning to rank challenge overview. In *Proceedings of the Yahoo! Learning to Rank Challenge, held at ICML 2010, Haifa, Israel, June 25, 2010*, pages 1–24.

Dai, Z., Xiong, C., Callan, J., and Liu, Z. (2018). Convolutional neural networks for soft-matching n-grams in ad-hoc search. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, WSDM '18, pages 126–134, New York, NY, USA. ACM.

Dehghani, M., Zamani, H., Severyn, A., Kamps, J., and Croft, W. B. (2017). Neural ranking models with weak supervision. In *Proceedings of The 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June. Association for Computational Linguistics.

Fan, Y., Guo, J., Lan, Y., Xu, J., Zhai, C., and Cheng, X. (2018). Modeling diverse relevance patterns in ad-hoc retrieval. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, SIGIR 2018, Ann Arbor, MI, USA, July 08-12, 2018*, pages 375–384.

Fuhr, N. (2018). Some common mistakes in ir evaluation, and how they can be avoided. *SIGIR Forum*, 51(3):32–41, February.

Guo, J., Fan, Y., Ai, Q., and Croft, W. B. (2016). A deep relevance matching model for ad-hoc retrieval. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, CIKM '16, pages 55–64, New York, NY, USA. ACM.

Guo, J., Fan, Y., Pang, L., Yang, L., Ai, Q., Zamani, H., Wu, C., Croft, W. B., and Cheng, X. (2019a). A deep look into neural ranking models for information retrieval. *CoRR*, abs/1903.06902.

[10]https://www.ncbi.nlm.nih.gov/pmc/

[11]LIG, Université Grenoble-Alpes

[12]School of Engineering and Physical Sciences, Heriot-Watt University, Edinburgh

Guo, J., Yixing, F., Xiang, J., and Xueqi, C. (2019b). Matchzoo: A learning, practicing, and developing system for neural text matching. In *Proceedings of the 42Nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR'19, pages 1297–1300, New York, NY, USA. ACM.

Hu, B., Lu, Z., Li, H., and Chen, Q. (2014). Convolutional neural network architectures for matching natural language sentences. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 2042–2050.

Järvelin, K. and Kekäläinen, J. (2002). Cumulated gain-based evaluation of IR techniques. *ACM Trans. Inf. Syst.*, 20(4):422–446.

Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Loper, E. and Bird, S. (2002). NLTK: the natural language toolkit. *CoRR*, cs.CL/0205028.

Mitra, B., Diaz, F., and Craswell, N. (2017). Learning to match using local and distributed representations of text for web search. In *Proceedings of the 26th International Conference on World Wide Web*, WWW '17, pages 1291–1299, Republic and Canton of Geneva, Switzerland. International World Wide Web Conferences Steering Committee.

Mizzaro, S. (1997). Relevance: The whole history. *JASIS*, 48(9):810–832.

Pang, L., Lan, Y., Guo, J., Xu, J., Wan, S., and Cheng, X. (2016). Text matching as image recognition. In *Thirtieth AAAI Conference on Artificial Intelligence*.

Pang, L., Lan, Y., Guo, J., Xu, J., Xu, J., and Cheng, X. (2017). Deeprank: A new deep architecture for relevance ranking in information retrieval. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, CIKM '17, pages 257–266, New York, NY, USA. ACM.

Pennington, J., Socher, R., and Manning, C. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Porter, M. F. (2001). Snowball: A language for stemming algorithms.

Robertson, S. E. and Walker, S. (1994). Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In *Proceedings of the 17th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval. Dublin, Ireland, 3-6 July 1994 (Special Issue of the SIGIR Forum)*, pages 232–241.

Sanderson, M. (2010). Christopher d. manning, prabhakar raghavan, hinrich schütze, *Introduction to Information Retrieval*, cambridge university press 2008. ISBN-13 978-0-521-86571-5, xxi + 482 pages. *Natural Language Engineering*, 16(1):100–103.

Sasaki, S., Sun, S., Schamoni, S., Duh, K., and Inui, K. (2018). Cross-lingual learning-to-rank with shared representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 458–463, New Orleans, Louisiana, June. Association for Computational Linguistics.

Schamoni, S., Hieber, F., Sokolov, A., and Riezler, S. (2014). Learning translational and knowledge-based similarities from relevance rankings for cross-language retrieval. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 488–494.

Schütze, H., Manning, C. D., and Raghavan, P. (2008). Introduction to information retrieval. In *Proceedings of the international communication of association for computing machinery conference*, page 260.

Urbano, J., Marrero, M., and Martín, D. (2013). A comparison of the optimality of statistical significance tests for information retrieval evaluation. In *The 36th International ACM SIGIR conference on research and development in Information Retrieval, SIGIR '13, Dublin, Ireland - July 28 - August 01, 2013*, pages 925–928.

Xiong, C., Dai, Z., Callan, J., Liu, Z., and Power, R. (2017). End-to-end neural ad-hoc ranking with kernel pooling. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '17, pages 55–64, New York, NY, USA. ACM.

Yang, W., Lu, K., Yang, P., and Lin, J. (2019a). Critically examining the "neural hype": Weak baselines and the additivity of effectiveness gains from neural ranking models. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2019, Paris, France, July 21-25, 2019*, pages 1129–1132.

Yang, Z., Dai, Z., Yang, Y., Carbonell, J. G., Salakhutdinov, R., and Le, Q. V. (2019b). Xlnet: Generalized autoregressive pretraining for language understanding. *CoRR*, abs/1906.08237.

Zamani, H., Dehghani, M., Croft, W. B., Learned-Miller, E., and Kamps, J. (2018). From neural re-ranking to neural ranking: Learning a sparse representation for inverted indexing. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 497–506. ACM.

Zheng, Y., Fan, Z., Liu, Y., Luo, C., Zhang, M., and Ma, S. (2018). Sogou-qcl: A new dataset with click relevance label. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, SIGIR '18, pages 1117–1120, New York, NY, USA. ACM.