AIR ASSIGNMENT 2 REPORT

GANELLARI ARTJOLA (12046001)
CSÉCSY, CSABA RÓBERT (11945931)
DASS, REEMA GEORGE (12144026)
TEMME JOHANNES (00306409)

1. Part 1 - FIRA Data to Test Collection Preparation

The annotated data belongs to three datasets: TREC Robust 04, DBPedia Entity, and TripClick. As the documents and queries led to more noisy results, we have to first provide a robust way of aggregating the raw judgements (multiple per pair) to labels (one grade per pair). In the data, we find that the judgements contain the user-id, time to annotate, time of day, the judgement value and the optional selection of text. In the data, we have 67,224 row entries identifiable by the triple of user, query and document (user judgement of a query-document pair). We refer to each of these entries as "observations".

When processing the data to finally aggregate the judgments based on majority vote or heuristic, we follow two consecutive steps:

(a) The annotator has to read the whole query and at least a certain percentage (10%) of the document to be able to judge its relevance to the query. If the time to annotate is too short in view of the characters that the annotator has to read, we will ignore the annotation.
(b) It is fine if annotators disagree sometimes with the majority vote of the other annotators, but if they disagree too often, we will ignore their annotations.

For step (a), we use the annotation time $t^{u,q,d}$ of a user u, query q and document d and compare it to the character lengths $len(q), len(d)$ of q and d. According to [1], one reads on average 987 characters per minute in the English language with a standard deviation of 118 characters. Assuming a normal distribution, we can say that 99.9% of all readers are able to read at most 1341 characters per minute in the English language.[1] This defines a global maximum reading speed v_{max} that we consider reasonable. We hypothesise that one has to read the full query as well as at least 10% of the document to judge whether it is relevant or not. When dividing the corresponding number of characters by the global maximum reading speed we get a minimum annotation time for a query-document pair:

$$\frac{len(q) + 0.1 \cdot len(d)}{v_{max}} = t^{q,d}_{min}.$$

According to our hypothesis, we exclude a query-document-pair of a user if the user's annotation time falls below the minimum annotation time, i.e., if $t^{u,q,d} < t^{q,d}_{min}$. In total, this concerns 6,366 observations (around 9.5%).

After removing queries according to step (a), we assess the inter-annotator agreement of step b via Cohen's Kappa. More specifically, for each user we compare the user's ratings with the majority judgements of all other users. If the majority judgement of a query-document pair is not unique (there are multiple modes represented in a list $r^{q,d}_{majority} = [r_1 < r_2 < r_3]$), we take the "middle[2] judgement" among them (i.e. $r^{q,d}_{majority} = r_2$). However, if the user's judgement $r^{u,q,d}$ is among the multiple majority

Date: June 2022.

[1]This is because a normal distribution covers around 99.9% of the probability mass in $[-\infty, \mu + 3 \cdot \sigma]$, where μ, σ denote the mean and standard deviation, respectively.

[2]if there are two or four modes, we consider the second or third mode as "middle judgement", respectively.

judgements (i.e. $r^{u,q,d} \in r^{q,d}_{majority} = [r_1 < r_2 < r_3]$, we take the user's judgement as majority judgement (i.e. $r^{u,q,d} = r^{q,d}_{majority}$).[3] With Cohen's Kappa, we then compare $(r^{u,q,d})_{q,d}$ with $r^{q,d}_{majority}$ for all query-document pairs q, d that user u judged. Averaged over all users, we observe a Cohen's Kappa of around 0.4 which is rather low and is commonly referred to as "weak agreement". We hypothesise that a user with Cohen's Kappa below 0.15 is extraordinary and too often contradicts with the majority vote of the other users. For that reason, we exclude users with Cohen's Kappa below 0.15. This concerns 3 users with in total 651 observations.

Note that we did not consider the other available information *time of day* or *optional selection of text* to process the judgements of query-document pairs. On the one hand, this is because we do not consider the time of day relevant for the annotation quality. On the other hand, we decided to ignore the optional selection of text, as text selection did not work on mobile browsers and did therefore not necessarily provide information on the quality of the annotation.

After excluding observations according to step (a) and (b), we aggregate the remaining judgements using a majority vote. If the majority is not unique for a query-document pair, we again use the majority rating that corresponds to the "middle judgement", as discussed above.

To test our preprocessing steps and the aggregated judgements, we randomly selected six query-document pairs, namely p_1 = (q_1 = trip_4003, d_1 =trip_11087377), p_2 = (trip_19422, trip_9336654), p_3 = (db_q_dbpedia:Luna_10, db_dbpedia:Luna_5), p_4 = (db_q_dbpedia:WLTN, db_dbpedia:WLTN), p_5 = (rob_qq_LA081689-0143, rob_LA020990-0032), p_6 = (rob_qq_LA082989-0095, rob_LA110489-0085). We now discuss the results of each pair individually:

(1) For the first pair p_1, we note that one of the two users that judged the pair was excluded due to a low Cohen's Kappa of 0.12. This user judged p_1 to be a perfect match (rating 3), while the second user marked it as only topic-related (i.e., 1). After analysing the query as well as the document, we would rather agree with the second user and consider the document topic-related but being too specific to be a perfect match for the broad query. As the first user was excluded due to a low Cohen's kappa, our preprocessing leads to an overall aggregated rating of 1, which we consider adequate.

(2) For the second pair p_2, two out of three users judged it too fast according to preprocessing step (a). Hence, the rating of the remaining user provided the overall judgement of 1. Admittedly, the two excluded ratings weren't too far off with 2 and 1, respectively. After assessing the document, we consider that the overall rating of 1 is fine.

(3) All three users of p_3 provided the same rating 3 in reasonable annotation time. Hence, the overall rating was 3, too. Indeed, the document fits perfectly to the query.

(4) Two users rate p_4 as a good match (i.e., 2), while the third user considers it a perfect match. As none of the ratings was excluded due to preprocessing steps (a) or (b), the overall rating was 2. After assessing the document, we would have considered a rating of 3 as more adequate. However, a rating of 2 is still close.

(5) For p_5, two out of three users gave rating 1, while the third user judged the document to be irrelevant to the query (rating 0), leading to an overall aggregated rating of 1. After assessing the document and the query, we would consider the document to be at most topic-related to the query. It covers the topics "marriage" and "Florida" separately, but does not treat "marriage in Florida". Overall, a rating of 1 seems to be acceptable.

(6) For the last pair p_6, we note that one user provided a too short answer according to preprocessing step (a). The other two users judged the pair to be a perfect (3) match and irrelevant (0),

[3]We follow this special rule to correctly calculate Cohen's Kappa. Otherwise, we would draw wrong conclusions on mismatches between the majority vote and the user's judgement.

respectively. Hence, the overall judgement was 3 according to our heuristic described above. After assessing the document in view of the query, we would consider both 0 and 3 inadequate. Instead, we would rather regard 1 to be a fair rating.

Overall, we note that the resulting judgements after aggregation seem to be tenable. Based on our sample, we also see that our exclusions according to preprocessing steps (a) and (b) are mostly adequate. Furthermore, we find that the minimum annotation time according to (a) was usually set at a conservative level. Hence, the majority of observations feature longer annotation times.

2. Discussion of Part 2

In this part, we were first required to implement three re-ranking models, convknrm, knrm and tk.

KNRM (Kernel-Based Neural Ranking-Model) is a simple model that does not use any kind of convolution, and the kernel does not use many learning parameters, only the embedding layer. In the implementation of this model, we have included also a fully connected linear layer with a bias to calculate the score based on the kernel values. Furthermore, we have ignored all query and document entries that are artificially made by paddings. Therefore, a matric multiplication has been made. Next, the cosine similarity between the query and document embedding is calculated, followed by the calculation of the kernel values based on the cosine similarity matrix.

The other model Conv-knrm which is an extension of the KNRM model,introduces convolutional neural networks into the process. This model considers convolutions of kernel size 1 to n-grams , i.e considers single words to n-grams. A list of 1-D convolution is created first, where the kernel of each convolution corresponds to the number of N-grams that we want to consider in the convolution. This is followed by the creation of the final linear layer used for scoring. It takes Kernel-Matrix as input and outputs a 1-D score for each kernel and for each N-gram pair of document and query, we get an input for the linear layer. After this, we had to ignore all query and document entries that are artificially made by paddings. After this, we transformed tensors in the required shape of [batch, emb-dim, sequence-length]. After matching the convoluted query tokens with the convoluted document tokens according to all possible combinations, we calculated the cosine similarity between the query and document embedding, followed by the calculation of the kernel values based on the cosine similarity matrix. Finally, we feed everything into the final linear layer to predict the score.

The final model, tk-model (Transformer-Kernel ranking model) is an interpretable lightweight, and effective model which combines transformers with kernel-pooling. This model can be formalized as follows, firstly starting with an encoding layer that uses transformers and does the encoding independently. Then, in the matching phase, it uses cosine similarity for each term by term interaction by creating in this way a single match matrix. Finally, kernel pooling is used which is similar to KNRM and counts the match-matrix interactions and then those interactions are summed up and weighted in the final scoring layer.

All of these models can be found in the ./src folder of this project where each of the respective files contains more detailed comments of the difficult parts of the model. In order to train and get results from these models, the 're-ranking.ipynb' has to be utilized and run. Inside this file, there is a json object called config. In the config, the chosen model can be specified. Before starting explaining the training and evaluation process, it is worth mentioning that the models explained above have been trained in the Google Colab Pro environment which provides a 24 GB NVIDIA Tesla K80 GPU.

Prior to the training process, early stopping has been included based on the validation set as it was required. It was interesting to notice that all the algorithms performed early stopping running only from Epoch 0 to Epoch 1. Working with a 'TRAINING-BATCH-SIZE = 256' we got the results in the validation set of the MSMARCO dataset as below:

MS-MARCO	KNRM	CONV-KNRM	TK
MRR@10	0.21700634920634898	0.2639878968253967	0.22363055555555525

In our case, in the validation set, the best performing algorithm was Conv KNRM with a MRR of 0.2639. The number of queries ranked in all the cases was 2000. Moreover, as it can be seen from the table below, Conv KNRM was the slowest algorithm to train, as it took more time to train than KNRM. Nevertheless, TK performed the training a lot faster than the other algorithms, and the results that it produced on the validation set are slightly better than KNRM.

Training	KNRM	CONV-KNRM	TK
Time(s)	8880	11358	5854

The results of the evaluation of the MS-MARCO dataset of the test set is as follows:

MS-MARCO	KNRM	CONV-KNRM	TK
MRR@10	0.21743452380952342	0.2646657744888344	0.22779702380952326

The results of the test set evaluation of the FiRA fine grained labels from part 1 and baseline label creation method are as follows:

Model	FiRA-2022 baseline label creation method	FiRA-2022 created labels from Part 1
KNRM	0.210129938334533	0.2033499588394
CONV-KNRM	0.261992889933399	0.2598447772288
TK	0.220122399959594	0.2130344498222

In general the results produced in both test datasets produced similar results. Similar to the validation set, 'Conv KNRM' performed better than the rest of the algorithms. Moreover, 'TK' performed slightly better than 'KNRM' which was also similar to the ranking in the validation set. Overall, all algorithms performed similar in both datasets. This goes to show that all of them generalize the function quite well and they are not prone to overfitting.

3. Discussion of Part 3 - Extractive QA

The Hugging Face Hub is a platform including over 50K models, 5K data sets, and 5K demonstrations that allows users to effortlessly cooperate on their machine learning workflows. The Hub is as a central location for anybody to exchange, explore, find, and interact with open-source Learning Algorithms.

You may find and utilize tens of thousands of open-source machine learning models contributed by the community. Model repositories are outfitted with Model Cards to alert users of each model's limits and biases in order to promote responsible model usage and development. Additional metadata regarding information such as their jobs, languages, and metrics can be published, and training metrics charts can even be included if the repository has TensorBoard traces. It's also simple to add an inference widget to your model, letting anyone to experiment with it right in the website! An API is given for stages of the design to quickly offer any model.

After understanding the process of Hugging face and its implementation we select a specific model and implement it in the following steps:

(a) Colab, or "Colaboratory," allows you to develop and run Python code in your browser.There is no setup required , GPUs are available for free and Simple sharing, Colab can help you whether you're a student, a data scientist, or an AI researcher.In only a few lines of code, you can import an image dataset, train an image classifier on it, and assess the model. Colab notebooks run code on Google's

cloud servers, allowing you to take use of Google technology, such as GPUs and TPUs, independent of the capabilities of your system. Colab can load public github notebooks directly, with no required authorization step.

(b) We then load data set provided that is, msmarco-fira-21.qrels.qa-answers, msmarco-fira-21.qrels.retrieval, and msmarco-fira-21.qrels.qa-tuples into the local variable to pass it to the model. Transformers provides APIs for conveniently downloading and training cutting-edge pretrained models. Pretrained algorithms may cut computation costs, minimize your carbon footprint, and save you time over training a model from scratch.

(c) We import the required models from Transformers library like RobertaTokenizer, RobertaForQuestionAnswering. While using these models from pretrained(model name) class method, a general tokenizer class will be instantiated as one of the library's tokenizer classes.

(d) We used the "roberta-base-squad2 model" for this exercise, which is the most popular model on Hugging Face. The model is based on the transformers model RoBERTa, and the SQuAD2.0 dataset was used for fine-tuning. It has been trained for the job of Question Answering on qestion-answer pairings, including unanswerable questions. For the pretraining of RoBERTa, only raw texts were used, meaning no human interaction was necessary for labelling. The objective of the training is Masked language modelling, which randomly masks 15% of the words randomly, then tries to predict them. The RobertaTokenizer is based on Byte-Pair Encoding tokenization, meaning that the words are splitted to characters, then merge rules are applied.

For step (d), We start with loading the model and the tokenizer in the code implemented in the extractiveqa python file and run the datasets through the model to receive the predictions for our data sets.

The Evaluation of the results are as follows: Regarding the answers to compare, the results are the same for the FiRA gold-label pairs and for the pairs resulting from the best re-ranking model. Computing F-statistics resulted in 0.32, while Exact matching resulted in 0.088. The model did not provide a prediction for about 1/5 of the questions. (There was no answer for 11971 questions out of 52606.)

4. Split of Work

Our team has split the workload of assignment 2 as follows:

(1) Ganellari Artjola:
- Part 2: implementation of `tk_model`, training process and result evaluation for MSMARCO dataset, early stopping, test set evaluation on MSMARCO and FiRA-2022 fine-grained labels on out-of-domain data
- Part 4: report writing for Part 2

(2) Csécsy, Csaba Róbert:
- Part 3: everything
- Part 4: small part of report writing for Part 3

(3) Dass, Reema George :
- Part 4: report writing of part 3

(4) TEMME Johannes:
- Part 1: everything
- Part 2: implementation of `knrm`, `conv_knrm`, training process (except for early-stopping), MRR result evaluation, MS Marco test evaluation
- Part 4: report writing for Part 1

References

[1] Susanne Trauzettel-Klosinski, Klaus Dietz, IReST Study Group, et al. Standardized assessment of reading performance: The new international reading speed texts irest. *Investigative ophthalmology & visual science*, 53(9):5452–5461, 2012.