RESEARCH REPORT

Exploring Knowledge-Graphs for clarification in product search

Research conducted at the

Center for Intelligent Information Retrieval of the

University of Massachusetts Amherst

from

David Arjuna Fingerlos

Salzburg, 13th October 2022

Contents

List of Figures

List of Listings

List of Tables

Research Questions

(RQ1) Can query aspects, candidate answers, and clarifying questions be encoded in a useful way that fosters KG and supports the subsequent clarification system?

(RQ2) How can a model be optimized and how does a model perform, that captures relationships within query aspects, candidate answers, and clarifying questions, with comparison to existing state-of-the-art clarification systems?

Supervision

The supervisor of this research project is assistant Professor in the College of Information and Computer Sciences (CICS) at the University of Massachusetts Amherst and Associate Director of the Center for Intelligent Information Retrieval (CIIR) Hamed Zamani[1]. As Supervisor from the home university acts Researcher and Lecturer at the Salzburg University of Applied Sciences Martin Uray[2].

Acknowledgement

This subsequential report is intended to serve as a foundation for the following Master thesis written by the author David Fingerlos. The thesis will be based on this topic and the research described in abbreviated form in this report. The topics will be augmented, which will result in the master thesis being a more comprehensive and extended version of the report. This means that all topics and results will be shared, and entire sections will be adapted accordingly.

[1] https://groups.cs.umass.edu/zamani/
[2] https://its.fh-salzburg.ac.at/ueber-uns/lehrende/detail/uray/

1 Introduction

This section is based on [1]. The World Wide Web is one massive knowledgebase but to extract any knowledge from it, it is necessary to search and find the desired information first. This is usually done through a search query, a word combination that describes what the user is looking for. Such search queries are often ambiguous, which makes it hard for the search engine to identify the actual intent. One approach of modern search engines for this problem is to diversify the represented results. This means to not only show the most probable search results for that query but also less likely results that may use a different interpretation of the query or use it in another context. For Example, when you google the query “car” you might find yourself with a list of results like the general definition, followed by a car rental company, followed by the website of a car manufacturer, and maybe also some ads for jobs in the car industry. Those results are quite diverse and vary strongly from each other, but this also means that there might be a matching result for any intended interpretation of the initial query. This approach works great for desktop applications and devices with bigger screens, but this approach has some limitations when using devices with a limited bandwidth interaction interface.

Nowadays every device is getting smart and thereby a lot of new form factors exist, and tech companies try to seamlessly integrate technology into our lives. Those new device types often have a unique way of interacting with them. Smart speakers, smart watches and smart fridges are only a few examples of connected devices, that have not been that technical a few years ago. The challenge with those new device types is, that the technology got integrated into an already existing device (speaker, watch or the classic fridge) and therefore it was necessary to design a new interface around the final product, to interact with the recently smart device. In some cases, those interfaces come with some limitations for example, the speaker just interacts via voice or with the watch to have a small screen. Those limitations have one thing in common, the potential information exchange is confined and therefore it is not possible to use the approach of diversification of the search result as it will just annoy the user if the smart speaker reads out 20 different search results or to see through such a long list of results on the small screen of a smartwatch.

Considering these factors, an alternative approach is required for devices with such a limited bandwidth interaction interface. The basic idea is to handle this situation as we humans naturally would do when we are confused or did not fully understand a request, to ask a question for clarification. This approach can also be used to narrow down a search query and quickly lead to the result the user was looking for.

When asking context-relevant questions that further specifies the exact input query, it is important to fully understand that query. Therefore, it is not only required to know what the search query means but also what the possible interpretations of those expressions and related topics are. All these requirements can be reached with a versatile trained knowledge graph.

2 Literature Review

In the following chapter, the technologies applied throughout this research project will be mentioned and explained more deeply. Furthermore, a preview of the modules used in the later-described experiments will be given.

2.1 Knowledge Graphs

The expression "Knowledge Graph" appears in literature since 1972, when Edward W. Schneider published his research on the application of course modularization [2].

One of the first appearances as the incarnation of how this phrase is used nowadays was in 2007 by DBpedia, a community effort, that resulted in a knowledge base trained on the structured information from Wikipedia [3].

In 2012 the Google Knowledge Graph was introduced, which combined data from a variety of sources to generate a knowledge base that connects simple strings to the things, which are expressed by the words. Thereby Google used the Knowledge Graph to give a search query a deeper understanding, and to see it not only as the word combination but as the entity it is and the relation it exhibits [4].

A Knowledge Graph is an abstraction of knowledge modelled as a graph. Such a representation provides a lot of benefits as graphs provide a structured representation of information which means that beyond the meaning, the relations to other entities are stored and thereby form the information domain as one connected construct rather than individual separated terms [5]. Hereinafter the definition Knowledge Graph is understood as a "graph of data intended to accumulate and convey knowledge of the real world, whose nodes represent entities of interest and whose edges represent potentially different relations between these entities" [5, p. 3].

2.1.1 Definition

Those semantic networks consist of a variety of entities, which are connected to other forms of this entity, the parent entities and child entities, that could be possible characteristics or properties of this entity. Furthermore, it has edges that represent relations to other entities. In general, a Knowledge Graph can be described as a composition of information entities and their relations to other information entities. One example of such a relation is shown in the Figure 1, where the entity "Manhattan" is connected via a "is in" relation to the entity "New York City".

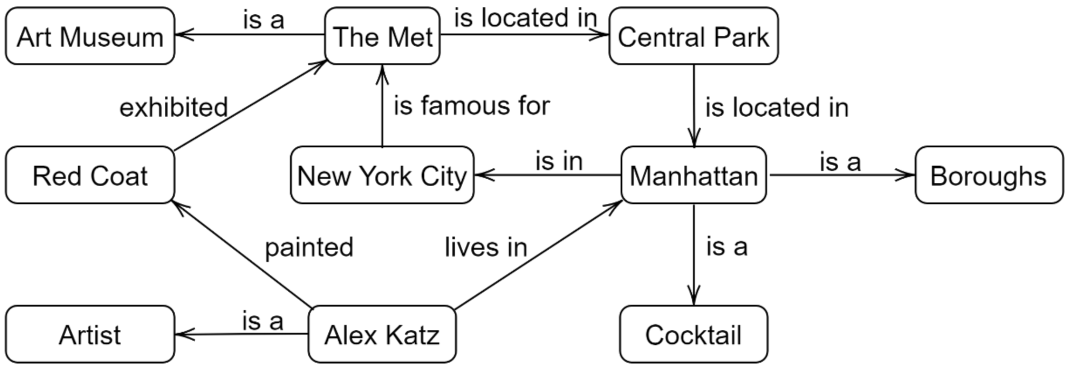


Figure 1 - Practical example of a Knowledge Graph

This form of construction graph can be applied in a variety of fields as shown in the picture above. With this example it can be understood that those structures are applied in the context of search engines as with the google knowledge graph. For instance, would the search query "Alex Katz" directly deliver the related information as his residence, profession, artworks, and further information which thereby can be used to find possible search results the user was looking for [5].

2.1.2 DBpedia

The following section is based on information from [6]–[10]. The core idea behind this community effort was to stitch all structured information available together to a big knowledge base, which then can be used to answer semantically rich queries. Up to then, this has only been done based on a specialized domain with a limited vocabulary as it generally was a problem to get enough relevant and structured information. To build such a domain-independent knowledge base, it is required to have a broad data corpus such as Wikipedia, which currently consists of over 6,2 Million articles in English. The resulting DBpedia knowledge base describes over 4.58 million things in the English version. DBpedia describes in over 140 Languages in total, more than 38 million things and approximately 20 billion triples[3]. The reason to train such a Knowledge Graph based on all the information – which already exists in a structured database – is, that with Wikipedia the search capabilities are limited to full-text search and thereby limits the possibilities to access that information base. DBpedia is not only trained-based on Wikipedia but is also interlinked with other open datasets. Furthermore, one of the core aspects of the development of DBpedia was to provide it with a variety of interfaces and different access modules so that the Knowledge base can be accessed over the web, via calls through web services and integrated into third-party

[3] https://dbpedia.org/page/N-Triples

applications. The DBpedia knowledge base can for example be requested through their web service DBpedia Lookup by using an URL, that simply holds the search query as a parameter.

2.2 Sequence-to-Sequence Models

The most prevalent sequence transduction models are advanced variations of recurrent neural networks (RNN) or convolutional neural networks (CNN). Those models usually consist of an encoder and a decoder part to convert the input sequence into the desired output sequence (Seq2Seq). Implementing an attention mechanism between those layers helps the model to learn what is important about the data and thereby to better understand the input, which leads to better-performing models [11]. Plain RNNs have problems with long-term dependencies as the gradient tends to vanish. Further development of the RNN is the long short-term memory (LSTM) model, that solves this vanishing gradient problem by introducing gates, that can detect important features in the data and carry those to a later stage. Thereby those models are capable of capturing long-distance dependencies in contrast to traditional recurrent units. In practice it has been shown, that for sequence-based tasks such LSTM models work well, especially with long-term dependencies, but the Transformer network architecture promises to be superior in quality and training time [12].

2.3 Transformers

The subsequential section is based on [11]. Those Seq2Seq Models usually consist of an encoder and a decoder part. Where the encoder maps the input sequence (x1,x2,…xn) onto a continuous representation z = (z1,z2,…zn). This representation is transformed through the decoder to the output sequence of symbols (y1,y2,…,ym). A lot of those models also use some sort of additional input in form of the previously generated symbol for generating the next one.

Generally, the Transformers consist of a similar architecture. On the encoder and the decoder side, there are point-wise multiplication, self-attention and fully connected layer stacked on top of each other, as shown in the Figure 2.

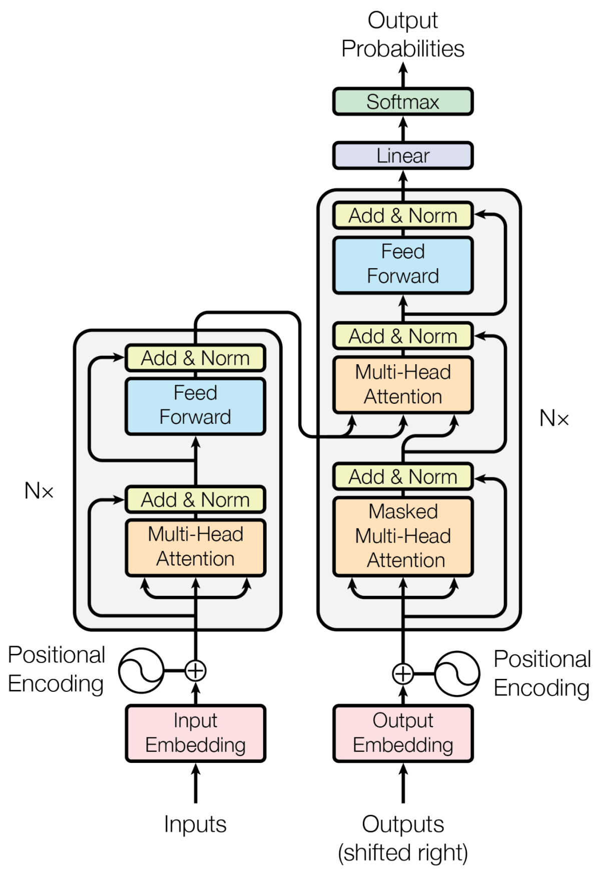


Figure 2 - The Transformer model architecture [13, Fig. 1].

The encoder consists of N times the identical layer, stacked together. This layer has a multi-head self-attention mechanism and a position-wise fully connected feed-forward network. On the decoder side, there is a similar structure but additionally to those two layers, you also find a multi-head attention layer that takes the output of the encoder stack as input. This multi-head attention performs an attention function on h times different learned projections of the queries, keys and values.

2.3.1 BERT

Bidirectional Encoder Representations from Transformers or short BERT, which use bidirectional learning to create state-of-the-art models for a wide range of tasks without modifying the architecture for one task specifically. A lot of the recent models are unidirectional and thereby restrict the possibilities of pre-trained approaches. Especially when applying fine-tuning to tasks like question answering, it is important to integrate context from both directions [14].

The approach of BERT consists of two steps the pre-training and the fine-tuning part. During pre-training unlabeled data is used to train the model. The obtained parameters from the pre-training step are applied as initialization during the fine-tuning. During this step, labelled data is used to fine-tune those parameters. The distinctive feature of BERT is, that it uses the same architecture for different tasks, as the same pre-trained model can be used to extract the initialization parameters for a variety of NLP tasks [14].

2.3.2 GPT2

The details described in this paragraph are originated in [15], [16]. As the successor to GPT, GPT-2 is a transformer-based language Model, that has been trained on a dataset of 8 million Web pages and uses 1.5 billion parameters to predict the next word in a text. Contrary to the usually applied supervised learning approach for such a natural language processing task, GPT-2 was trained without any explicit supervision. This means the Model was only trained on the raw texts without any labelling or human input at all. Because of the diversity of the used training dataset, the GPT-2 Model works well across a variety of domains and can be applied in many downstream tasks such as summarization, translation, reading comprehension, and question answering. Even though this model is quite versatile and can be applied to a variety of tasks, it does not manage to match the state-of-the-art scores with those downstream tasks but this Model suggests, that the usage of unsupervised techniques might be beneficial. The capabilities of the Model are shown where GPT-2 outperforms other language models, that are trained specifically on that one domain like Wikipedia, without being trained on that domain-specific dataset.

2.3.3 BART

This Model is a denoising autoencoder used to pre-train sequence-to-sequence models. The training mechanism behind BART consists of two steps. In the first step, the model is using an arbitrary noising function to corrupt the input text. This is followed by the second step, which aims to reconstruct the original text for the noised input. The Model behind this setting is composed of a Transformer-based machine translation architecture with the influence of many recent pretraining schemes like BERT because of the bidirectional encoder and GPT due to the usage of a left-to-right decoder. The used noising mechanism combines random shuffling, that reorders the original sentence and a filling-in scheme to replace text parts. As shown in Figure 3, this resembles the BERT part (1) of the BART Model where the Bidirectional Encoder replaces randomly tokens with masks and the missing tokens are independently predicted but generation tasks are difficult with that approach. Contrary to this, the GPT part (2) uses auto-regressive prediction to enable generation tasks but only works in a leftward context, which means that bidirectional interactions cannot be learned with such a model. The symbiosis of these two components results in the BART Model (3) where the Output of the Encoder is used as input to the Decoder part.

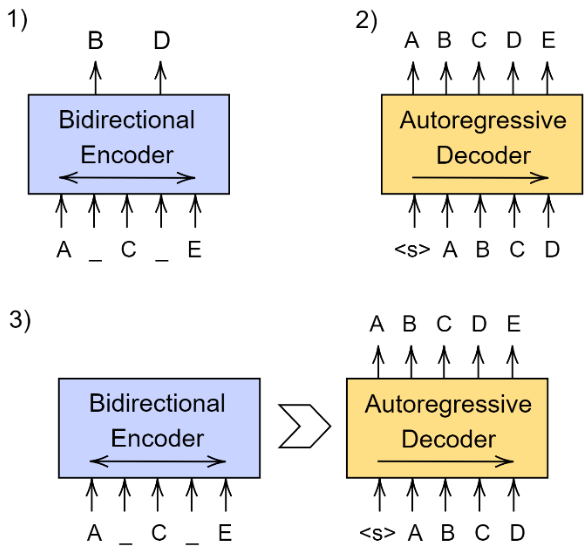


Figure 3 - BART: the schematic composition of BERT and GPT

BART is especially effective in the application of a text-generation task. This model can achieve new state-of-the-art results on tasks like question answering, summarization and abstractive dialogue and match the performance of RoBERTa on SQuAD and GLUE. For further details about this model refer to paper [17] which also serves as reference source for this section.

2.4 Evaluation Metrics

Those Metrics are essential for evaluating the results of the prediction model and comparing them to the ground truth data. The ensure diversity throughout the evaluation a variety of scores and variations of those scores are applied during the results evaluation part.

2.4.1 BLUE

The BLUE or Bilingual Evaluation Understudy was originally developed for translation but can also be used for text generation evaluation. This score represents the similarity between two input texts and can therefore be used to compare texts [18]

2.4.2 ROUGE

The Recall-Oriented Understudy for Gisting Evaluation Score is a text comparison metric, that counts the overlapping n-grams between two texts. Therefore, it can be utilized to determine the similarity between the ground truth and the generated clarification question [19]

2.4.3 BERTscore

This automatic evaluation metric is a similarity score for text generation. This score calculates a similarity score for each token of the ground truth and the generated question. With this score, contextual embeddings take place instead of exact matches, which is done by many other score matrices [20]

3 Methodology

In this following section, the techniques and scripts for the further proceeding will be discussed. Furthermore, it will be examined what kind of structure is used during training and prediction of the experiments conducted.

3.1 Hugging Face

The open-source pre-trained model library Hugging Face will be used as the foundation for the subsequently finetuned BART Model. Hugging Face has an extensive collection of Transformer models for a variety of tasks like Image Classification, Translation or Text Generation. Those pre-trained models can be implemented and quickly fine-tuned for a specific application, which makes the training process of the model a lot easier and faster [21].

3.2 Dataset

The foundation for the following work and the emerging research results is the Dataset. As the general goal of this research project was to explore clarification questions, it is necessary to use a dataset, that resembles this task within the data gathered. Furthermore, it would be beneficial to have a broad dataset with a huge amount of data points to different question domains as this helps to train robust and universal models.

MIMICS[4] is a large-scale data collection for search clarification, that has been conducted as asking clarification questions has become one of the main aspects of conversational information-seeking systems. This data collection is based on Bing search logs of real-world search queries, that have been augmented with a clarification question and up to five candidate answers for this question. This dataset has been developed by the supervisor to this project Hamed Zamani in cooperation with Microsoft for specifically such a task as generating clarification questions [22].

MIMICS consists of three datasets:

MIMICS-Click contains 400k unique queries with clarification questions and corresponding candidate answers.

MIMICS-ClickExplore represents 60k queries of exploration data.

[4] https://github.com/microsoft/MIMICS

MIMICS-Manual consists of 2k real queries, that have been manually labelled by a team of at least three trained annotators.

Those Datasets combined have a broad spectrum of data points regarding clarification questions and represent the foundation for subsequent efforts. Since a lot of data is required for training, the MIMICS-Click dataset with its 400k queries will be used as input for the Train-Test-Split and therewith for the training process [22].

3.3 Data Analysis

For the initial step before the preprocessing or even training of a model, it is necessary to conduct a comprehensive data analysis for a better understanding of the dataset and the search domain.

A detailed analysis of the input data has revealed that over 95% of the 414361 queries in the MIMICS-Click dataset contain a clarification question with the expression "Select one to refine your search". This means, that in those cases it is suggested to refine the search by adding the candidate's answer as further information rather than using it to clarify a question. But on the other side this means, that those queries are useless for further experiments on this behalf because for training a model and furthermore to generate clarification questions, it is required to have such a question as training input rather than a generic phrase without any information content. Without those queries, that are just used as refinement, there are 19228 queries with real questions left in the dataset.

Moreover, there are a lot of generic questions like "which 'dni' do you mean?" as this abbreviation is known as 'director of national intelligence' but can also be used as 'distributable net income'. Those 7485 generic questions also have no further information content, which leads us to 11743 queries that can be used for training the model.

3.4 Data Preprocessing

During the Preprocessing phase, in the first step, those redundant queries are going to be removed from the dataset. Additionally, it is favorable to filter the Dataset since we do not need all fields provided and therefore, we can reduce the amount of data, that needs to be processed later on. To simplify the data handling the filtered dataset was stored containing just the columns query, target and candidate answers where the target is the ground truth clarifying question and the candidate answers are the possible answers concatenated to a string.

3.5 Pipeline

Having diverse real-world queries with a ground truth clarifying question and candidate answers we try to use the query as input for the Knowledge Graph to extract some further details and specific facets about that query. These facets provide additional information about that topic or subject described in the search query. The extracted facets are then used together with the query as input for the trained model to generate a corresponding clarifying question. Those components compiled as one Pipeline are represented in the flowchart in Figure 4, where it is also shown which information is passed on between those components. This flowchart represents just the logical structure to visualize the procedure between the dedicated components to get from one input query to the generated clarifying question. Further details of the components, the technologies used for each component and the entire process of how to train and set up all those parts will be examined in detail in the following chapter.

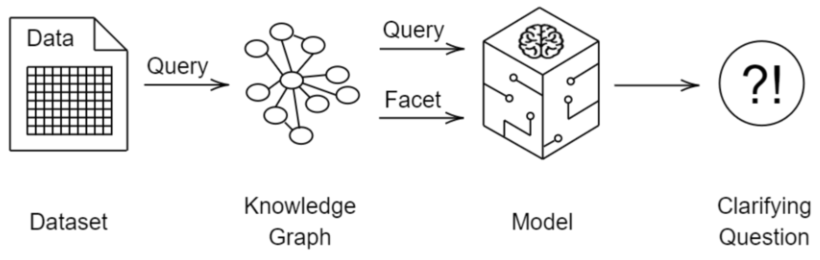


Figure 4 - Pipeline Logical Structure

3.5.1 Training-Pipeline

The entire procedure to generate a clarification question consists of several components, that all serve their individual purpose but need to work together as the subsequential component relies on their output. The beginning marks the MIMICS-Click Dataset. For the next step by dealing with the Knowledge Graph, only the queries are required as we want to extract from the graph further information about that input search query. In this phase, the Knowledge Graph DBpedia is used as the source of knowledge about that input query. More specifically the API Implementation DBpedia Lookup[5] is requested to enquire further details. The output of a DBpedia Lookup provides us with further information about the input query as this is shown in the Listing 1.

[5] https://github.com/dbpedia/lookup

```
<ArrayOfResults>
  <Result>
          <Label>Manhattan</Label>
          <URI>http://dbpedia.org/resource/Manhattan</URI>
          <Description>Manhattan (), often referred to by residents of the New York
                       City area as the City, is the most densely populated …
          </Description>
          <Classes>
                  <Class>
                          <Label>Settlement</Label>
                          <URI>http://dbpedia.org/ontology/Settlement</URI>
                  </Class>
                  <Class>
                          <Label>Place</Label>
                          <URI>http://dbpedia.org/ontology/Place</URI>
                  </Class>
                  …
          </Classes>
          <Categories>
                  <Category>
                          <URI>…River_islands_of_New_York_(state)</URI>
                  </Category>
                  <Category>
                          <URI>…County_seats_in_New_York_(state)</URI>
                  </Category>
                  …
          </Categories>
          …
  </Result>
  …
</ArrayOfResults>
```

Listing 1 - DBpedia Lookup Response

The response from the DBPedia knowledge base contains a list of most likely results and besides the label to the searched query and a description it also has corresponding classes and categories. These information classify and categorize the searched topic. By adding the classes, which the search query is associated with and the categories it belongs to, the query is further described. For example, can be differed between Manhattan the "settlement" and Manhattan the "cocktail with whisky" by adding those facets to the query. As categories and classes both have further details about that searched label but are still a bit different in the meaning, we crawl both information to the query for now and will compare what combination of those works best for the question generation task.

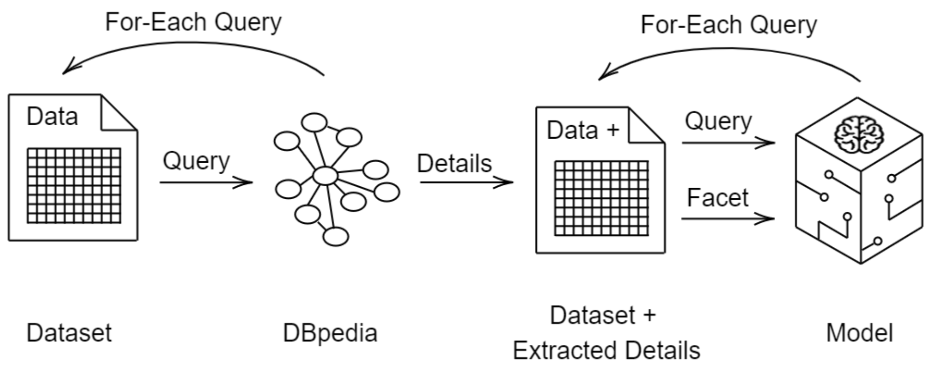


Figure 5 - Training Pipeline

For a later real-world demonstration the entire pipeline will work with one query as input and then calculate the clarifying question individually for every request. For the development and training of the Model to generate the Clarifying question it is necessary to have now the entire dataset prepared with the exact information as it would be used in a functional approach. The logical structure of the pipeline during the training process is shown in Figure 5. To get the classes and categories for all 11743 queries, a crawler script was used to request the Knowledge Graph of DBpedia and extract the three most likely classes and categories for the first five results. This crawler script can be shown in Listing 2. Thereby it is ensured that the information input of the search query is quite diverse but still focused on the topic as just the five most related results to the topic are used.

```
…
for idx, item in df.iterrows():
        url = base_url + item['query']

        #request DBpedia for that query
        response = callFascetGenerationAPI(url)

        #parse the response
        res_content = xmltodict.parse(response.content)

        #extract the categories and classes
        categories.append(' , '.join(extractXCategories(res_content, 5)))
        classes.append(' , '.join(extractXClasses(res_content, 5)))
 …
```

Listing 2 - DBpedia Crawler Script

With this crawled information added to the dataset, the resulting trainings-dataset consists of the query, target (clarifying question) and a list of classes and categories similar to the structure represented in Table 1. Those are structured in two columns with the query separated by a separation token from the list of classes or categories and as target the desired clarifying questions.

query	target	classes	categories
rimouski	What do you want to know about the place?	Settlement , City , Place , Administrative Region , Place , Populated Place ...	Cities and towns in Quebec , Populated places on the Saint Lawrence River , Populated places established in 1696 , Quebec federal electoral districts ...
ubac	Which "ubac" do you mean?	Settlement , Place , Populated Place , Organisation , Sports Team , Agent , Organisation , Soccer Club ...	1977 establishments in the United Kingdom , Defunct trade unions of the United Kingdom , Finance sector trade unions , Municipalities of Bohol ...
book of ezra	What would you like to know about this book?	Book , Written Work , Work , Award , Person , Writer , Agent	Ezra–Nehemiah , Ketuvim , Texts in the Septuagint , 1st-century BC books ...
freiberg disease	What do you want to know about this condition?	Disease , Plant , Species , Eukaryote , Disease , Person , Agent , Scientist ...	Osteonecrosis , Lists of compositions by composer , Compositions by Karl Michael Ziehrer , Hybrid grape varieties ...

Table 1 - Augmented Dataset with DBpedia Response

3.6 Model to Train

The training script for the NLP model was developed with PyTorch and Huggingface based on the described dataset to learn how to generate the desired output. As pre-trained model, the *facebook/bart-base*[6] was chosen, which was implemented with the corresponding tokenizer and adapted to train a BART Model for generating clarifying questions based on the query and some facets as input. As shown in the Listing 3, a Pytorch Lightning[7] Model (LitModel) was used as a deep-learning framework for training the BART model.

```
…
#initialize the BART tokenizer and model
tokenizer = BartTokenizer.from_pretrained('facebook/bart-base', add_prefix_space=True)
bart_model = BartForConditionalGeneration.from_pretrained("facebook/bart-base")

#load the dataset into the tokenizer for training
summary_data = SummaryDataModule(tokenizer, base_dir + 'MIMICS_cla_train.csv',
                                 batch_size = 64, num_examples = 20000)

#initialize the model with the data
model = LitModel(learning_rate = 2e-5, tokenizer = tokenizer,
                 model = bart_model, hparams = hparams)

#initialize the trainer
checkpoint = ModelCheckpoint(dirpath=base_dir)
trainer = pl.Trainer(gpus = 1,
                     max_epochs = 10,
                     min_epochs = 5,
                     auto_lr_find = True,
                     checkpoint_callback = checkpoint, progress_bar_refresh_rate =
    100)
```

[6] https://huggingface.co/facebook/bart-base
[7] https://www.pytorchlightning.ai/

```

#start the training
trainer.fit(model, summary_data)
 …
```

Listing 3 - Trainingsscript

During the training process, two different types of models were developed. The first variation of the Model was trained based on the query combined with the classes extracted from the DBpedia response for that query as input and the clarifying question as the desired result. For the second variation the query with the extracted categories was used.

3.7 DBpedia Pipeline

The centre of the pipeline represents the previously trained clarifying question generation model in the respective version. The procedure of the pipeline starts with the input of a search query. In the next step, the DBpedia Knowledge Graph is requested for this search query. From the response to that request, the facets are extracted and passed on to the model with the initial query. The pre-trained model is then used to determine the clarifying question. The structure of this Pipeline based on DBpedia can be seen in the Figure 6.

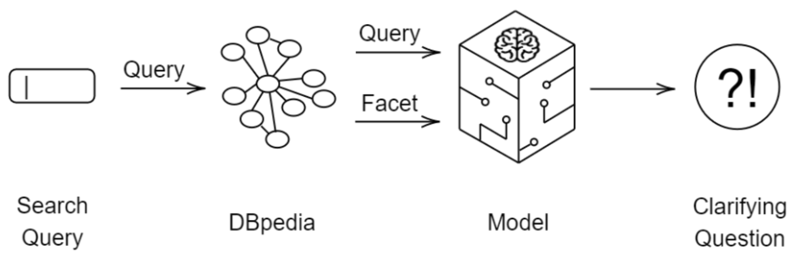


Figure 6 - DBpedia Pipeline

The pipeline was constructed in a way, that the Model itself and the input generation module are loosely connected modules in the entire pipeline and can easily be exchanged. This helps by being able to use this entire pipeline for different experiments without the need to adapt multiple sections.

3.8 Alternative Approach to Information Gathering

To draw an objective assessment of the significance and to give those obtained results relevance it is necessary to have comparable results of an alternative approach. Without any acknowledged state-of-the-art baselines it is hard to draw a conclusion about the relevance of the results, generated by the pipeline, based on information from the DBpedia Knowledge Graph. DBpedia is an extremely versatile and highly developed Knowledge Graph with its

20 billion triples. As it is not possible to build and train a comparable knowledge graph without a massive computational infrastructure and in such a limited time, further proceeding was discussed with the supervisor of this research project Hamed Zamani. After considering the possible options, it was agreed to develop a different approach to obtain comparative metrics and to focus therefore on the facet extraction and generation framework called Faspect[8], which was recently developed and published by the PhD student in Information Retrieval at the CIIR Chris Samarinas. The information in this section are from the sources [23], [24].

3.8.1 Faspect

This section is based on information from the published paper to this facet generation toolkit [23]. Faspect is an open source developed toolkit for facet extraction and generation for an open domain. The facets, generated by this toolkit, are based on five different models: Facet Extraction as Sequence Labeling, Autoregressive Facet Generation, Facet Generation as Extreme Multi-Label Classification, Facet Generation by Prompting Large Language Models and Unsupervised Facet Extraction from SERP (for further details on those different models see [23]). The facets extracted and generated by those models are combined and output as a list. Furthermore, there are three different ranking methods implemented in this framework: round-robin, MMR and rank. The implementation of this Framework to generate facets for the pipeline as a comparative approach to the DBpedia knowledge graph, was discussed with the author of the toolkit. For the implementation it was recommended to use the default settings and parameters like in the published paper. Furthermore, it was discussed what kind of results each of the five models generates, and which of them seems favorable for this specific task. It was determined that the results of the Unsupervised Facet Extraction might be too abstract for this specific generation task and therefore this model was excluded from the later implementation. The other four models should deliver quite topic-focused facets and still provide some diversity in the types of facets generated.

The Faspect implementation can be requested via an integrated API, however, for performance reasons and as simplification, the framework was directly implemented into the base code from the input generation module of the pipeline. The Faspect Framework requires a search query and some documents that describe the query. A possible input is shown in the Listing 4. The Faspect framework uses these input query and documents to generate the facets as shown in Listing 5. Those documents need to consist of detailed information and

[8] https://github.com/algoprog/Faspect

texts, which describe the query or related topics. This is required because the models of the Faspect framework need any context and knowledge to determine what this requested query is about and to extract or generate possible facets accordingly. One example of such a document would be the page content of a result provided by a search engine since search engines crawl the world wide web for documents that match the desired query.

Input:

```
{
    "query": "bicycle",
    "documents": [
                    "A bicycle is also called a pedal cycle, cycle or bike…",
                    "Types of Bicycles: BMX, Mountain Bike, Cruiser…",
                    …
                    "Document n text"
                ]
}
```

Listing 4 - Faspect Input

Possible Output:

```
{
    "facets": [
                  "helmet",
                  "bikes for kids",
                  "bmx",
                  "amazon",
                  "sale",
                  "for men",
                  "for women",
                  …
              ]
}
```

Listing 5 - Faspect Output

Those required inputs represent a further challenge for the training step as the dataset provided does not contain any further texts about the corresponding topic. But it is also required for a later pipeline implementation to have a method that extracts extensive describing texts about that topic specified. Therefore, a script was developed that uses the search engine DuckDuckGo[9] to extract some documents related to the provided query. This script sends a search request with, that desired query through DuckDuckGo and extracts the URLs to the first recommended webpages. In the next step, those web pages are called, and the content of that page is crawled. As that raw content consists of HTML tags and unwanted spaces and tabs because of the different formatting of those webpages, it is required to apply some extensive preprocessing to the crawled webpage content. At first, the HTML tags are

[9] https://duckduckgo.com/

deleted, and the excessive characters are removed. After that, the characters are UTF-8 decoded and the different snippets get stitched together into the final document. This process gets repeated for all web pages provided by the search engine as top results to that search query. Using the Faspect Toolkit as input facet generation leads to the pipeline structure, as shown in the Figure 7.

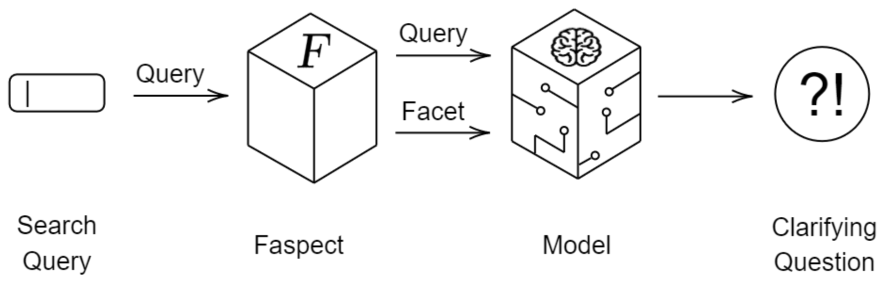


Figure 7 - Faspect Pipeline

3.9 Facets and DBpedia

With those two approaches to get some further details about that search query in form of facets implemented, it is also possible to combine them. The idea behind this will be, that the Faspect script results in a simple list of related keywords to the search query and that maybe even more information about the searched topic could lead to a better prediction of the clarification question. As the DBpedia extraction mechanism is already implemented and works stand-alone with just some input query, it would be possible to combine those two methods. This would mean using every facet as an input query to request the DBpedia knowledge graph about it, extract the classes or categories to every facet and thereby extend the facet from just some strings, that describe the search query to an input where those describing facets are specified in more detail. A possible result of the combination of the Faspect framework with the DBpedia Knowledge Graph is represented in Listing 6.

```
Bicycle </s> helmet = safety, headgear, cycling equipment | bikes for kids = training
wheels, running, riding | bmx = cycling, bicycle motocross…
```

Listing 6 - Combination of Faspect and DBpedia

3.10 Web Interface

All those separated scripts, frameworks and libraries are combined into the pipeline which can be run as one component and opens its API interface to request this system. This API can simply be called with just a query as the input parameter. Since it is quite cumbersome to set up all the packages and the proper Python environment locally just to check the results of the clarification question generation pipeline, a Web interface[10] was implemented in addition to the API. This Webpage runs on a web server of the College of Information and Computer Sciences with an entire python runtime environment, set up with all the packages and tools required to run the pipeline. It provides a simple user interface with a search bar to put in the desired query. By submitting the query, all the stages of the pipeline run through and the clarifying question, as well as the possible facets to clarify this question will be prompted as a response to the submitted request as shown in Figure 8.

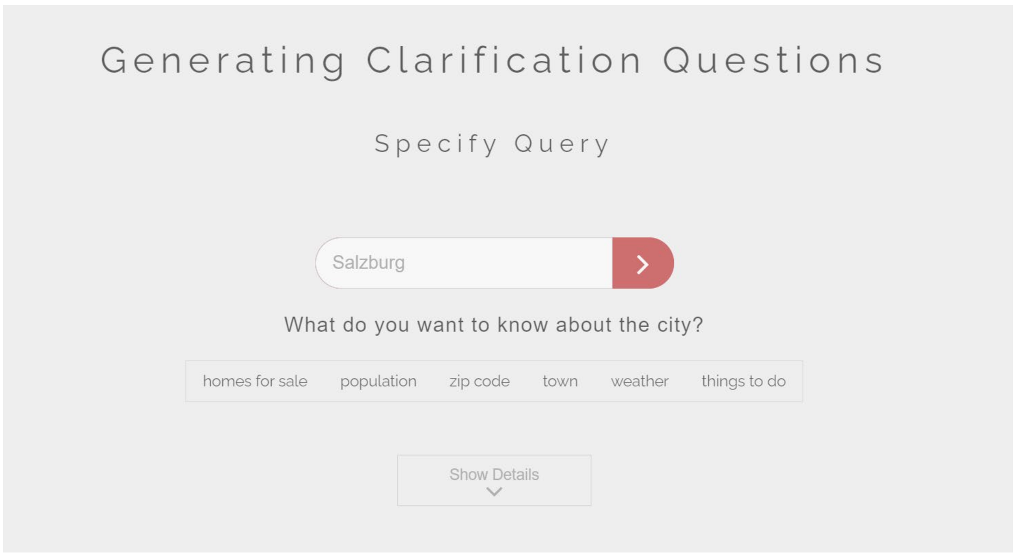


Figure 8 - Web Interface

[10] http://goulburn.cs.umass.edu/

4 Experiments and Results

In the following chapter, the techniques and approaches discussed in the previous section are applied to the dataset through the pipeline training structure. The used dataset is the MIMICS Click, which is split into 90% training data and 10% evaluation data for the training process.

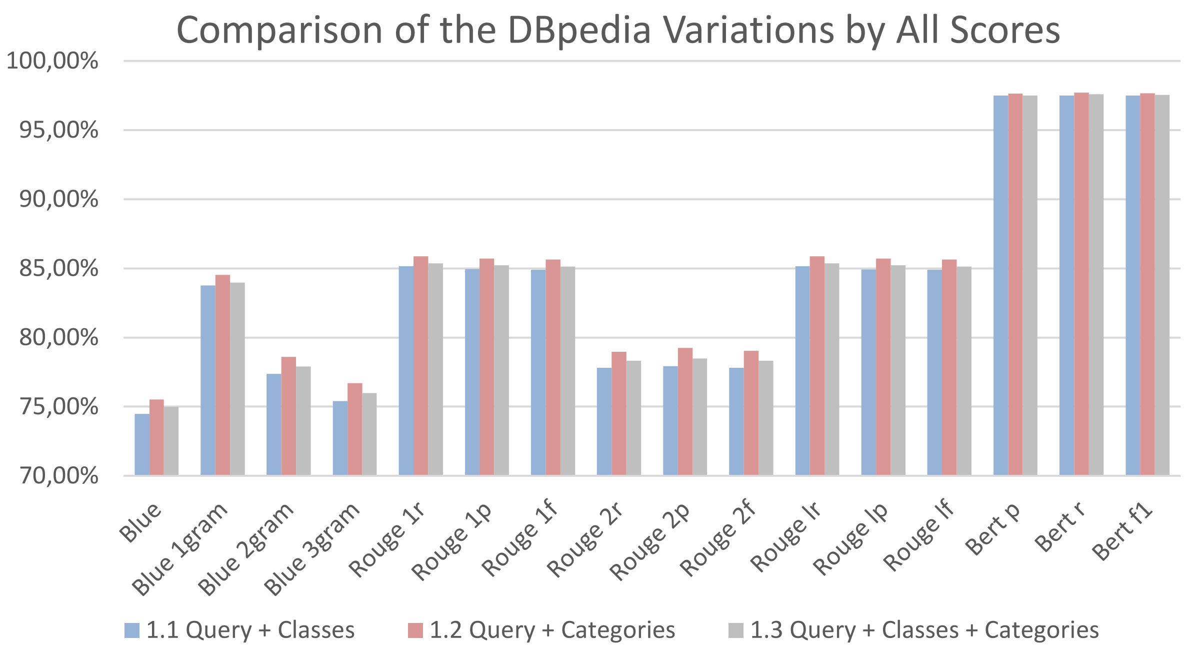
4.1 DBpedia

In the first experiment we trained the model with the query as input, augmented by extracted classes from DBpedia. In the second run we then used the same setup but with the extracted categories instead of the classes. As a comparison, we also completed a third run with the classes and the categories concatenated together and separated by a separation token, as presented in the Listing 7.

```
1.1 Query + Classes
clifton community school | Organisation , School , Educational Institution …
1.2 Query + Categories
clifton community school | School districts in British Columbia , Education in Surrey, British
Columbia …
1.3 Query + Classes + Categories
clifton community school | Organisation , School , Educational Institution … | School
districts in British Columbia , Education in Surrey, British Columbia …
```

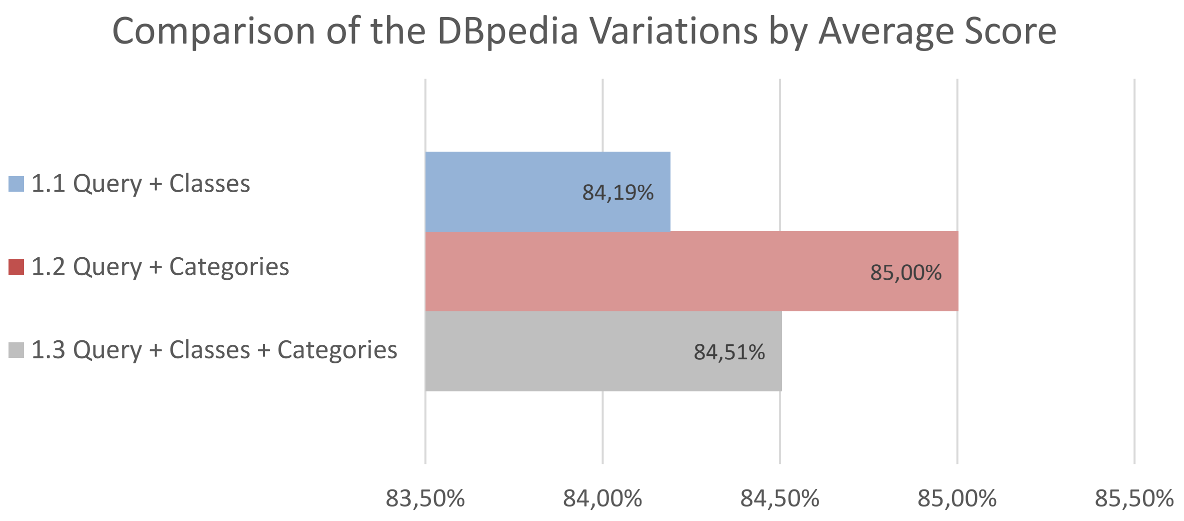
Listing 7 - Model-Input Variations of DBpedia

After the training of the clarifying question generation model, the evaluation dataset was used to test the effectiveness of the trained model. The model was used to predict for all the queries of the evaluation dataset, the clarifying question, and this generated clarifying question was then compared to the ground truth question. Based on those results, the Blue, Rouge and Bert scores were applied to determine the accuracy of the match for all queries. This comparison is represented in Graph 1.



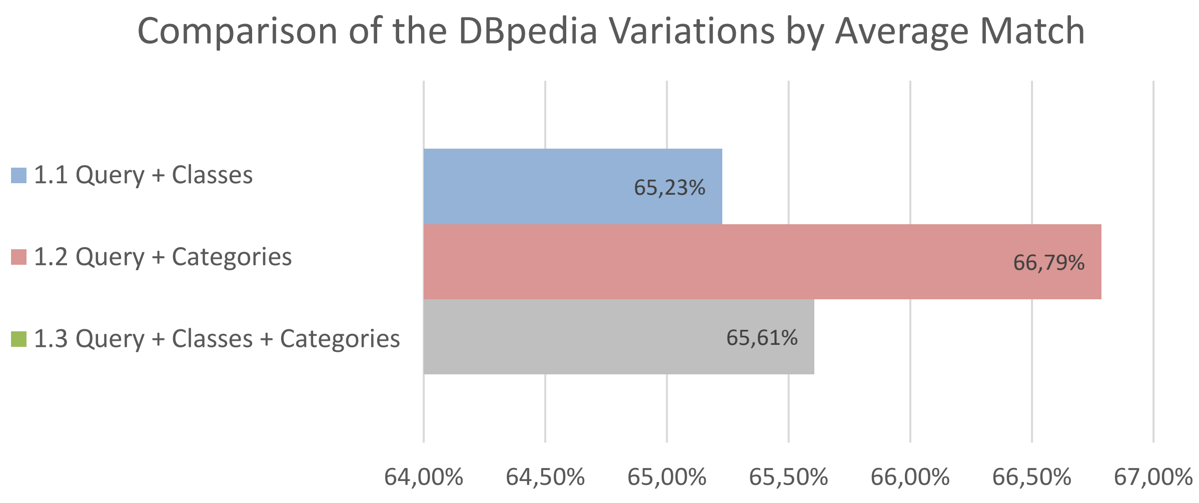
Graph 1 - Comparison of the DBpedia Variations by All Scores

As seen in the representation above, the Model trained with just the classes works the worst followed by the Model, that uses both the classes and the categories combined. The Model, that just uses the Categories beside the query as input, works the best with this setup. This characteristic can be observed for every score metric and because of that, the average overall scores were calculated and represented in the Graph 2.



Graph 2 - Comparison of the DBpedia Variations by Average Score

The average score represented above shows, that all three types of input data enable a good model as all scores are between 84,19% and 85%, which is quite a fine margin. But it is interesting, that for the evaluation part of the training dataset the categories alone deliver a better result than the categories with the classes combined. In comparison to the average score metrics, also the count of the perfect matches between the ground truth clarifying question and the generated one was calculated, as shown in Graph 3. Moreover, these scores follow the same trend and with the perfect match quote between 65,23% and 66,79% which implies, that the models perform decently for that dataset.



Graph 3 - Comparison of the DBpedia Variations by Average Match

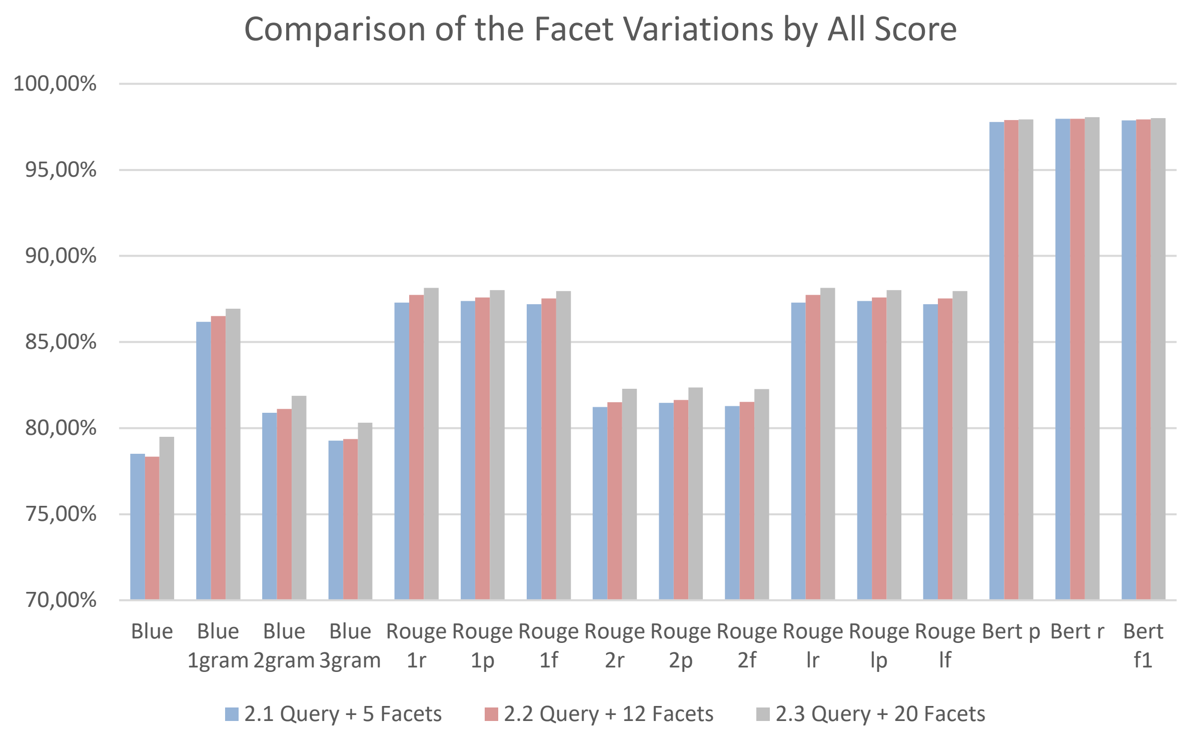
4.1.1 Interpretation

These results can be interpreted to mean that the categories hold more useful information for the clarifying question generation, compared to the classes extracted from the DBpedia results. Furthermore, additional information does not always mean better results as all classes combined with the categories lead to slightly worse scores. This might be because the model has too much information to consider than just to generate that one phrase and as with more information, there might also be more diversity, which leads to less focus on the more relevant facets. All in all it is impressive, how accurate those clarifying questions can be generated with just the query text and some facets related to that topic.

4.2 Faspect

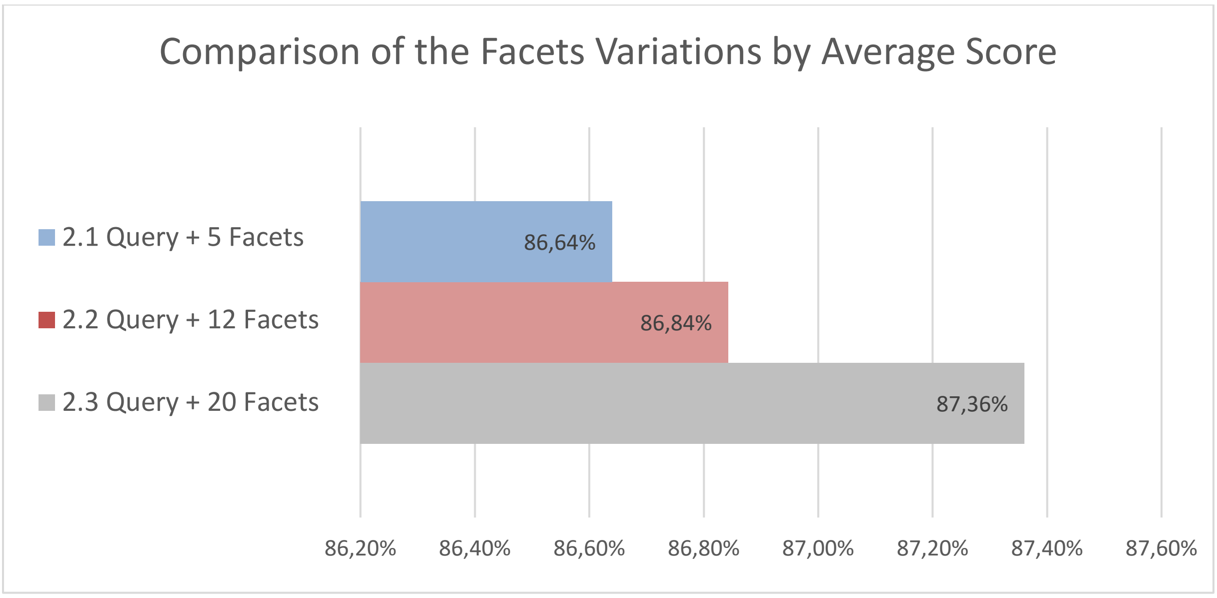
The implementation was done as discussed with the author of the framework. Therefore, the framework code was adopted in a way, that the unsupervised facet extraction model was excluded, and the remaining four models randomly aggregate their facets together through round-robin like it is proposed in the published paper. A question for optimization with facet extraction is, how many facets of the generated output should be used to achieve the best performance. The Framework outputs 20 facets of all those 4 models combined, but as seen in the previous experiment, can too much information also lead to poor performance of the generation model. To determine what works best for the facet extraction in combination with the clarification question generation model, it was tried to train a model with 5, 12 and with all 20 used facets.

As shown in Graph 4 where the achieved results are represented, it seems like most of the metrics follow the same trend where the score increases with the number of facets used. But as also recognizable with the previous experiment is the margin between those three models quite minimal.



Graph 4 - Comparison of the Facet Variations by All Score

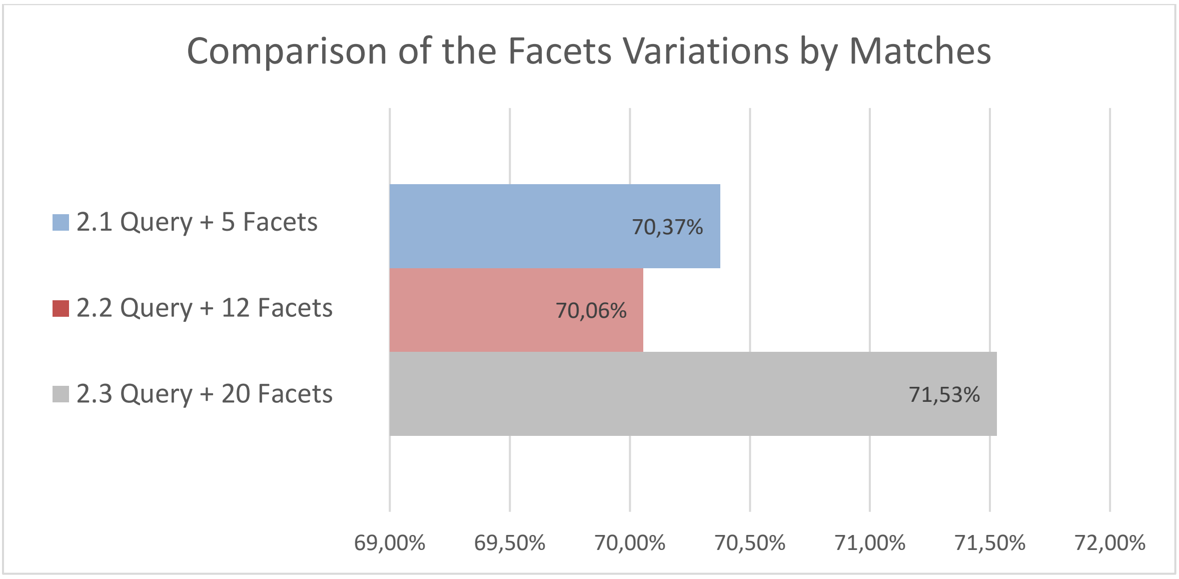
By comparing the average of the scores between these three models it can be seen that the difference between the scores is marginal with just 0,72% between 5 and 20 facets. But the training of the clarification question generation model with the input of the Faspect framework works quite good as all trained models achieve above 86% as the average score metric.



Graph 5 - Comparison of the Facets Variations by Average Score

4.2.1 Interpretation

The result of the average score metric shown in Graph 5 compared to the perfect match quote of the generated clarification questions as represented in Graph 6, reveals some interesting observations. The percentage value of the model trained with 12 facets is below the one with 5 facets and on the other hand, the value for the run with 20 facets is above the other two. Based on these developments it seems like the model with the 12 facets can in total generate more precise questions than that model with 5 facets, but those questions are not always exactly the same as the ground truth one. Like in some cases the first model generates more perfect matches, but the second model can in general adapt better to the question domain and create a more precise question for that query. The reason for this observation can be, that this model gets more input information about that query and because of that, it is more spread around the topic and because of the less topic specificity in fewer cases, the same query is generated.



Graph 6 - Comparison of the Facets Variations by Matches

Regardless of that, the Model trained with the 20 Facets as input delivers the best values with the highest average score and perfect match quote. This result can be understood in the way, that those huge amounts of facets about that query can be used by the model to learn and determine which of the facets are relevant and only consider them to precisely predict a relevant clarifying question. But these results also suggest that the model better understands what kind of information is irrelevant, would just spread the results around the exact topic and thereby not take them into account for the decisions of the final clarification question.

4.2.2 Comparison

In comparison to the results achieved with DBpedia, the different metrics follow the same trend but are a bit elevated compared to the facet extraction. The average scores increased from around Ø84,6% to around Ø86,9%. More drastic is the change with the quote of the perfect matches, where the model trained based on DBpedia achieved around Ø66% and the Model based on the Fascpect fascets hits around Ø70,8%.

To get a feeling for how the model works and what the numbers mean, a quick comparison of some randomly selected queries is represented subsequently. For this comparison, 10 queries were selected and 7 of them were removed as those were just 100% matches in both cases and thereby not interesting for any comparison. In Table 2 there are the results for 3 queries represented with the first line (marked with I) being the query separated to the first 5 facets in the format as it was input for the trained model. In the second line (marked with II) the first column holds the target question, which is the annotated ground truth and would have been used to train the model with this query. This one is followed by the predicted one of the trained model, and the average score over all the calculated score metrics for those two questions. The first line, marked with a light grey background, is always the input and result of the DBpedia approach based on the categories trained model. The second line holds the results from the Faspect-based variation with the usage of all 20 facets. For both cases, the model with the best results of the respective variant was deliberately used for this comparison. The full data can be found in the appendix at the end of this report.

I = **Query | Facets**

II = **Target / Predicted / Avg. Score**

I	didanosine | Hepatotoxins , Purines , Nucleoside analog reverse transcriptase inhibitors , EC 3.6.1 , RNA …		
II	What would you like to know about didanosine?	What would you like to know about didanosine?	100,00%
I	didanosine | didanosine fda approval , side effects of dosing , side effects , men , didanosine package insert …		
II	What would you like to know about didanosine?	What do you want to know about this medication?	50,38%

I	lowe syndrome | Syndromes affecting the eyes , Amino acid metabolism disorders , Syndromes affecting the kidneys , Syndromes affecting the heart , Cardiac arrhythmia …		
II	What do you want to know about this condition?	What do you want to know about this medical condition?	89,91%
I	lowe syndrome | lowe syndrome adults , inheritance , diagnosis , men , lowe syndrome in children …		
II	What do you want to know about this condition?	What do you want to know about this condition?	100,00%

I	electric vacuum pump | Vacuum pumps , Oil wells , Pumps , Irrigation , Vacuum …		
II	What device are you looking for?	What do you want to do with electric vacuum pump?	28,08%
I	electric vacuum pump | dishwasher , ebay , master power , refrigerator , amazon …		
II	What device are you looking for?	What appliance are you looking for?	76,37%

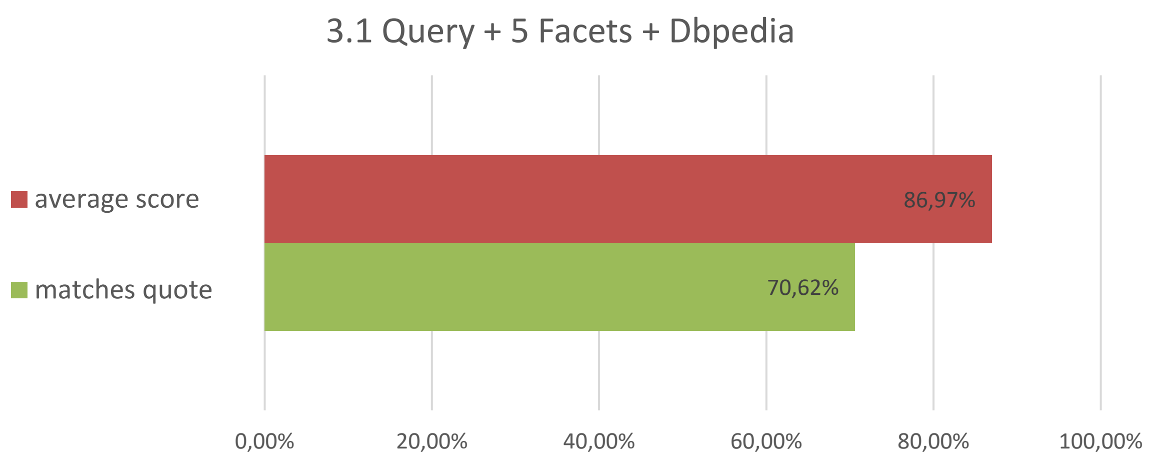
Table 2 - Comparison of Faspect and DBpedia

It is interesting to see the results of the computed clarification question in comparison to the ground truth one. For example, in the first query where the scores are worse for the Faspect variation, the model was able understand, that didanosine is a medication and phrased the question more naturally but also correctly to ask ‘this medication’ instead of ‘didanosine’ directly. In the second case DBpedia did it also right but just concretized the formulation by clarifying it as ‘medical condition’. With the third example it can be seen, how those two models just phrased different questions where the Faspect-based approach better resembles the meaning of the ground truth question but the DBpedia one is a different interpretation of what the user could ask for with this query.

Like in these examples, it would be very interesting to let some people manually annotate and rank the predicted results as the metrics do not always tell the truth regarding the understanding or meaning of a question to that query. But because of the short amount of time during this research period and the limited resources, it is not possible to conduct a survey on this topic and we must rely on those metrics, which anyway do a decent job for most of the cases.

4.3 Facets and DBpedia Combined

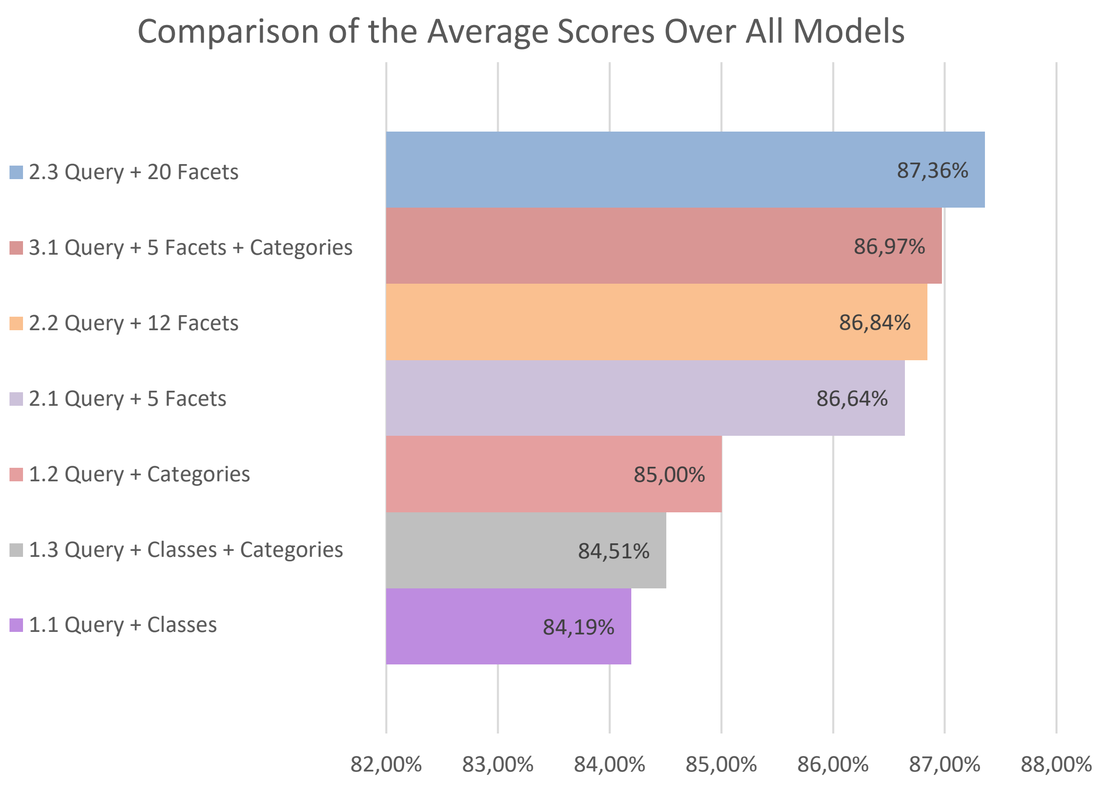
As this approach quickly dramatically increases the amount of the input data, it was determined to just use the Faspect variant with the 5 facets per query. As seen with the results of this approach, represented in Graph 7, do the percentages show an increase over the use of the 5 Facets model without the DBpedia expansion.



Graph 7 - 3.1 Query + 5 Facets + Dbpedia

4.4 Comparison of All Experiments

In the Graph 8, the comparison between all these approaches with their different variations is shown. It can be determined, that the Faspect approach generally works a bit more precisely according to the metrics than the approach based on DBpedia. Especially interesting is the augmentation of the 5 facets where the categories from DBpedia lead to an increase of the average score, resulting in a higher value than the model with 12 Facets did achieve.



Graph 8 - Comparison of the Average Scores Over All Models

All in all, the results above show that every variation tried, did deliver enough information so the model can predict a clarifying question for most cases. On the one hand, does the Faspect approach deliver the best results but on the other hand, does this approach need an extra script to call the web for input documents. That crawler uses the search result from a search engine and the most relevant ranked pages to download the content and extract the required documents. In comparison, the approach with DBpedia relies on a sophisticated knowledge graph that can easily be requested with just one call and does not need to crawl the results of another search engine. With all these simplifications the DBpedia approach delivers only marginal worse results, which still seem sufficient for most applications and considering the extra efforts required for the Faspect variation, the pipeline variation based on DBpedia might be overall the better approach.

5 Research Questions

(RQ1) Can query aspects, candidate answers, and clarifying questions be encoded in a useful way that fosters KG and supports the subsequent clarification system?

Yes, as the conducted research examined in this report shows, it is possible based on the trained knowledge stored in a knowledge graph to support a system for clarification question generation. To achieve a representative result for a broad domain with this approach it is required to have a knowledge graph that is trained for a variety of topics with enough diversity and versatility to also extract some background information of related topics even though the Knowledge base was not trained with that particular information. A well-trained Knowledge Graph can be a very versatile instrument to support the subsequential system with its encoded information.

(RQ2) How can a model be optimized and how does a model perform, that captures relationships within query aspects, candidate answers, and clarifying questions, with comparison to existing state-of-the-art clarification systems?

The output quality of a model trained for asking clarification questions depends on the information it receives as input about the domain and query. Therefore, it is required to deeply analyze what the best source for knowledge is in that specific domain, the system is intended for. The most acknowledged clarification systems are assessed based on manual evaluation by a group of specially trained and qualified judges. As such an evaluation was not possible, a comparison to such systems cannot be drawn. But for the realization of this clarifying question generation system state-of-the-art models and technologies have been used and the achieved results look promising based on the scores calculated for the comparison of the ground truth clarifying question with the generated one. As shown, the developed system is able to achieve in most cases, the exact target clarification question, and in a big percentage of the remaining cases a quite similar one which leads to an average score for the best system variation over 87%. The similarly good results achieved with two different approaches suggest, that the selected BART model is suited and can be well-trained for handling the task. The quality evaluation used in this research report is based on a variety of NLP metrics that assess the similarity between two texts. The combination of comparison metrics used throughout this framework is new in that extend as the evaluation of a clarification question generation system. Therefore, this report and more specifically the

Master thesis, which will be elaborated as an augmented and extended version of this research, can be seen as the new baseline for future systems in that field of research and thereby understand as the current state-of-the-art.

6 Resume

In conclusion to the astonishing results achieved through the research on this project, enabled through the Austrian Marshall Plan Foundation scholarship, I am very thankful to get this opportunity. The period abroad was not only great to deeply integrate into that research topic but also helped in getting to know another work and study culture. In addition to that, it was enriching to gain some experience in the research department, especially in such a highly qualified research facility. It helped a lot that the CIIR at the University of Massachusetts Amherst, where I conducted my research, is specialized in information retrieval and thereby at the core topic of my research project. Moreover, it has facilitated the intentions of the project, that my supervisor recently published some papers on his research about related topics and is thereby fully aware of the current state-of-the-art techniques and possible approaches for this task. Furthermore, it was unbelievable helpful that the Dataset used as the fundament for this research was developed by the supervisor himself. This meant, that he knew every detail about it and that made it a lot easier to get on with the data analysis and focus on the development of the core research topic. All in All, I have to say that my stay and research at the Center for Intelligent Information Retrieval of the College of Information and Computer Sciences at the University of Massachusetts Amherst was a great experience and an incredibly valuable enrichment to my university education. I want to say a special thank you to the Austrian Marshall Plan Foundation, my two supervisors and all the colleagues and partners that helped me and made this possible.

[1] H. Zamani, S. T. Dumais, N. Craswell, P. N. Bennett, and G. L. Microsoft, “Generating Clarifying Questions for Information Retrieval,” 2020, doi: 10.1145/3366423.3380126.

[2] E. W. Schneider, “Course Modularization Applied: The Interface System and Its Implications For Sequence Control and Data Analysis,” *ERIC*, 1973.

[3] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives, “DBpedia: A nucleus for a Web of open data,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 4825 LNCS, pp. 722–735, 2007, doi: 10.1007/978-3-540-76298-0_52/COVER.

[4] “Introducing the Knowledge Graph: things, not strings.” https://www.blog.google/products/search/introducing-knowledge-graph-things-not/ (accessed Oct. 18, 2022).

[5] A. Hogan *et al.*, “Article 71.-mermann. 2021. Knowledge Graphs,” *ACM Comput Surv*, vol. 54, no. 4, 2021, doi: 10.1145/3447772.

[6] “dbpedia/dbpedia-lookup: A generic entity retrieval service for linked data. Contains presets to replicate the DBpedia Lookup service.” https://github.com/dbpedia/dbpedia-lookup (accessed Oct. 18, 2022).

[7] “Largest encyclopedia online | Guinness World Records.” https://www.guinnessworldrecords.com/world-records/85651-largest-encyclopedia-online (accessed Oct. 18, 2022).

[8] Julia Holze, “DBpedia Snapshot 2022-03 Release - DBpedia Association,” May 02, 2022. https://www.dbpedia.org/blog/dbpedia-snapshot-2022-03-release/ (accessed Oct. 18, 2022).

[9] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives, “DBpedia: A nucleus for a Web of open data,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 4825 LNCS, pp. 722–735, 2007, doi: 10.1007/978-3-540-76298-0_52/COVER.

[10] Daniel Fleischhacker, Volha Bryl, and Christian Bizer, “DBpedia Version 2014 released - DBpedia Association,” Sep. 09, 2014.

https://www.dbpedia.org/blog/dbpedia-version-2014-released/ (accessed Oct. 27, 2022).

[11] A. Vaswani *et al.*, "Attention is All you Need," in *Advances in Neural Information Processing Systems*, 2017, vol. 30. [Online]. Available: https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

[12] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling," Dec. 2014, doi: 10.48550/arxiv.1412.3555.

[14] J. Devlin, M.-W. Chang, K. Lee, K. T. Google, and A. I. Language, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," *Proceedings of the 2019 Conference of the North*, pp. 4171–4186, 2019, doi: 10.18653/V1/N19-1423.

[15] "Better Language Models and Their Implications." https://openai.com/blog/better-language-models/ (accessed Oct. 18, 2022).

[16] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language Models are Unsupervised Multitask Learners", Accessed: Oct. 18, 2022. [Online]. Available: https://github.com/codelucas/newspaper

[17] M. Lewis *et al.*, "BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension," pp. 7871–7880, Jul. 2020, doi: 10.18653/V1/2020.ACL-MAIN.703.

[18] "A Gentle Introduction to Calculating the BLEU Score for Text in Python." https://machinelearningmastery.com/calculate-bleu-score-for-text-python/ (accessed Oct. 18, 2022).

[19] C.-Y. Lin, "ROUGE: A Package for Automatic Evaluation of Summaries," in *Text Summarization Branches Out*, Jul. 2004, pp. 74–81. [Online]. Available: https://aclanthology.org/W04-1013

[20] T. Zhang, V. Kishore, F. Wu, K. Q. Weinberger, and Y. Artzi, "BERTSCORE: EVALUATING TEXT GENERATION WITH BERT", Accessed: Oct. 18, 2022. [Online]. Available: https://github.com/Tiiiger/bert_score.

[21] "Hugging Face – The AI community building the future." https://huggingface.co/ (accessed Oct. 18, 2022).

[22] H. Zamani, G. Lueck, E. Chen, R. Quispe, F. Luu, and N. Craswell, "MIMICS: A Large-Scale Data Collection for Search Clarification," *International Conference on Information and Knowledge Management, Proceedings*, pp. 3189–3196, Oct. 2020, doi: 10.1145/3340531.3412772.

[23] C. Samarinas, A. Dharawat, and H. Zamani, "Revisiting Open Domain Query Facet Extraction and Generation", doi: 10.1145/3539813.3545138.

[24] J. Lehmann *et al.*, "DBpedia - A large-scale, multilingual knowledge base extracted from Wikipedia," *Semant Web*, vol. 6, no. 2, pp. 167–195, 2015, doi: 10.3233/SW-140134.

7 Appendix

1.3 DBpedia + Categories + Classes

	Blue	Blue 1gram	Blue 2gram	Blue 3gram	Rouge 1r	Rouge 1p	rouge 1f	rouge 2r	rouge 2p	rouge 2f	rouge lr	rouge lp	rouge lf	bert p	bert r	bert f1	Average
#Matches	1244	1244	1244	1244	1254	1261	1244	1244	1244	1244	1254	1261	1244	1260	1265	1267	**1251**
%Matches	65,23%	65,23%	65,23%	65,23%	65,76%	66,12%	65,23%	65,23%	65,23%	65,23%	65,76%	66,12%	65,23%	66,07%	66,33%	66,43%	**65,61%**
Average score	75,00%	83,98%	77,90%	75,99%	85,36%	85,22%	85,14%	78,32%	78,48%	78,33%	85,36%	85,22%	85,14%	97,50%	97,59%	97,54%	**84,51%**

1.1 DBpedia + Classes

	Blue	Blue 1gram	Blue 2gram	Blue 3gram	Rouge 1r	Rouge 1p	rouge 1f	rouge 2r	rouge 2p	rouge 2f	rouge lr	rouge lp	rouge lf	bert p	bert r	bert f1	Average
#Matches	1238	1238	1238	1238	1248	1249	1238	1238	1238	1238	1248	1249	1238	1255	1253	1258	**1244**
%Matches	64,92%	64,92%	64,92%	64,92%	65,44%	65,50%	64,92%	64,92%	64,92%	64,92%	65,44%	65,50%	64,92%	65,81%	65,70%	65,96%	**65,23%**
Average score	74,48%	83,76%	77,38%	75,40%	85,16%	84,94%	84,90%	77,82%	77,92%	77,80%	85,16%	84,93%	84,89%	97,50%	97,50%	97,50%	**84,19%**

1.2 DBpedia + Categories

	Blue	Blue 1gram	Blue 2gram	Blue 3gram	Rouge 1r	Rouge 1p	rouge 1f	rouge 2r	rouge 2p	rouge 2f	rouge lr	rouge lp	rouge lf	bert p	bert r	bert f1	Average
#Matches	1266	1266	1266	1266	1275	1280	1266	1266	1266	1266	1275	1280	1266	1290	1289	1295	**1274**
%Matches	66,39%	66,39%	66,39%	66,39%	66,86%	67,12%	66,39%	66,39%	66,39%	66,39%	66,86%	67,12%	66,39%	67,64%	67,59%	67,90%	**66,79%**
Average score	75,52%	84,53%	78,61%	76,69%	85,87%	85,71%	85,65%	78,97%	79,25%	79,04%	85,87%	85,70%	85,64%	97,63%	97,70%	97,66%	**85,00%**

2.3 Faspect + 20 Facets

source	didanosine | didanosine fda approval , side effects of dosing , side effects , men , didanosine package insert , side effects of a generic , pancreatitis , women , for use , infection.it , infection															
target	What would you like to know about didanosine?															
predicted	What do you want to know about this medication?															
metric	**Blue**	**Blue 1gram**	**Blue 2gram**	**Blue 3gram**	**Rouge 1r**	**Rouge 1p**	**rouge 1f**	**rouge 2r**	**rouge 2p**	**rouge 2f**	**rouge lr**	**rouge lp**	**rouge lf**	**bert p**	**bert r**	**bert f1**	**Average**
score	0,00%	55,16%	25,21%	14,71%	62,50%	55,56%	58,82%	28,57%	25,00%	26,67%	62,50%	55,56%	58,82%	90,42%	94,32%	92,33%	**50,38%**

source	lowe syndrome | lowe syndrome adults , inheritance , diagnosis , men , lowe syndrome in children , living with , lowe syndrome association , women , mutation , treatment , causes , genetics															
target	What do you want to know about this condition?															
predicted	What do you want to know about this condition?															
metric	**Blue**	**Blue 1gram**	**Blue 2gram**	**Blue 3gram**	**Rouge 1r**	**Rouge 1p**	**rouge 1f**	**rouge 2r**	**rouge 2p**	**rouge 2f**	**rouge lr**	**rouge lp**	**rouge lf**	**bert p**	**bert r**	**bert f1**	**Average**
score	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	**100,00%**

source	electric vacuum pump | dishwasher , ebay , master power , refrigerator , amazon , men , air conditioner , jegs electric vacuum pump , refrigerant , 12 volt , 12 , accessory , mcmaster															
target	What device are you looking for?															
predicted	What appliance are you looking for?															
metric	**Blue**	**Blue 1gram**	**Blue 2gram**	**Blue 3gram**	**Rouge 1r**	**Rouge 1p**	**rouge 1f**	**rouge 2r**	**rouge 2p**	**rouge 2f**	**rouge lr**	**rouge lp**	**rouge lf**	**bert p**	**bert r**	**bert f1**	**Average**
score	53,73%	83,33%	60,00%	50,00%	83,33%	83,33%	83,33%	60,00%	60,00%	60,00%	83,33%	83,33%	83,33%	98,30%	98,30%	98,30%	**76,37%**

1.2 DBpedia + Categories

source	didanosine | Hepatotoxins , Purines , Nucleoside analog reverse transcriptase inhibitors , EC 3.6.1 , RNA , Nucleotides , EC 3.6.1 , Living people , 1950 births , HIV/AIDS researchers , Protein domains															
target	What would you like to know about didanosine?															
predicted	[' What would you like to know about didanosine?']															
metric	**Blue**	**Blue 1gram**	**Blue 2gram**	**Blue 3gram**	**Rouge 1r**	**Rouge 1p**	**rouge 1f**	**rouge 2r**	**rouge 2p**	**rouge 2f**	**rouge lr**	**rouge lp**	**rouge lf**	**bert p**	**bert r**	**bert f1**	**Average**
score	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	**100,00%**

source	lowe syndrome | Syndromes affecting the eyes , Amino acid metabolism disorders , Syndromes affecting the kidneys , Syndromes affecting the heart , Cardiac arrhythmia , Syndromes , Esophagus disorders , Vomiting , Human anatomy , Human back , Hematology , RTT , Coagulopathies , Respiratory therapy , Respiratory diseases															
target	What do you want to know about this condition?															
predicted	[' What do you want to know about this medical condition?']															
metric	**Blue**	**Blue 1gram**	**Blue 2gram**	**Blue 3gram**	**Rouge 1r**	**Rouge 1p**	**rouge 1f**	**rouge 2r**	**rouge 2p**	**rouge 2f**	**rouge lr**	**rouge lp**	**rouge lf**	**bert p**	**bert r**	**bert f1**	**Average**
score	79,56%	89,48%	78,30%	76,70%	100,00%	90,00%	94,74%	87,50%	77,78%	82,35%	100,00%	90,00%	94,74%	99,60%	98,63%	99,11%	**89,91%**

source	electric vacuum pump | Vacuum pumps , Oil wells , Pumps , Irrigation , Vacuum , Rocket engines using kerosene propellant , Rocket Lab rocket engines , Space programme of New Zealand , Automotive engineering , Automotive steering technologies , Vehicle safety technologies , ExxonMobil buildings and structures , Companies based in Milwaukee , National Register of Historic Places in Milwaukee , Semiconductor device fabrication															
target	What device are you looking for?															
predicted	[' What do you want to do with electric vacuum pump?']															
metric	**Blue**	**Blue 1gram**	**Blue 2gram**	**Blue 3gram**	**Rouge 1r**	**Rouge 1p**	**rouge 1f**	**rouge 2r**	**rouge 2p**	**rouge 2f**	**rouge lr**	**rouge lp**	**rouge lf**	**bert p**	**bert r**	**bert f1**	**Average**
score	0,00%	17,11%	0,00%	0,00%	33,33%	22,22%	26,67%	0,00%	0,00%	0,00%	33,33%	22,22%	26,67%	90,36%	88,16%	89,25%	**28,08%**