Computational Linguistics in the Netherlands Journal 12 (2022) 83-97 Submitted 12/2022; Published 12/2022

BERT-based Transformer Fine-tuning for Dutch Wikidata Question-Answering

Niels de Jong[*] NAJONG99@HOTMAIL.NL
Gosse Bouma[*] G.BOUMA@RUG.NL

[*] *University of Groningen, Groningen, the Netherlands*

Abstract

People rely on data to understand the world and inform their decision-making. However, effective access to data has become more challenging over time: data has increased in volume and velocity, as has its variability in truthfulness, utility, and format. Therefore, improving our interfaces to data has become a pressing issue. One type of interface has lately gained renewed attention, driven by advances in artificial intelligence: natural language interfaces. As of yet, though, improvements in natural language processing (NLP) have largely concentrated on English. Thus, we propose a text-based Dutch question-answering (QA) interface for accessing information on Wikidata (URL: `https://www.wikidata.org/`), driven by a Dutch-to-SPARQL BERT-based transformer model. Said transformer is a type of encoder-decoder model characterised by use of self-attention. In our application, it is trained to accept sentences in Dutch and to transform these into corresponding SPARQL queries. By subsequently evaluating the obtained queries at a knowledge base, users can retrieve answers to their questions. Since our model learns end-to-end, we need to train it using a dataset consisting of pairs of Dutch questions and SPARQL queries. To this end, we closely follow the procedure of Cui et al. (2021). Particularly, we create a Dutch machine-translated version of LC-QuAD 2.0 (Dubey et al. 2019) and apply entity and relation masking on the NL inputs and SPARQL outputs for increased generality, producing a dataset with 2,648 examples. We then let the transformer model fine-tune on the training subset of this dataset, using system-level BLEU score as the performance measure. Our final transformer configuration obtains a test BLEU score of 51.86, which seems to be in line with results found by Cui et al. (2021). Additionally, we conduct a qualitative analysis of our model's outputs, focusing especially on situations where the predicted SPARQL queries are incorrect. Here, we observe that queries involving infrequently-used SPARQL keywords and queries containing literals prove challenging to the transformer, as sometimes do the syntax of SPARQL and the general length of queries. Finally, we conclude our paper by proposing some potential future directions for our Dutch QA system.[1]

1. Introduction

We live in the era of 'Big Data', characterised by the high-volume storage and high-throughput exchange of information along with variation in kind, truthfulness and value of said information (Jain 2016). Since data plays an integral role in multiple state-of-the-art technologies such as computer vision and speech recognition, the Big Data era holds significant expectations. Still, realising these expectations requires us to find ways to effectively process the new influx of data, which is an ongoing process.

As data is often stored in databases (DBs), many of the challenges surrounding Big Data concentrate on the interactions with DBs, which are mainly performed via queries. One of the problems with such queries is that they are formulated in specific querying languages (QLs) that may be challenging to learn for non-expert users.

The querying issue can be avoided by providing more accessible end-user interfaces (UIs). Before the Big Data era, this was often successfully done via graphical (GUIs) or form-based UIs

1. The source code that accompanies this article can be found at `https://github.com/some-coder/dutch-kbqa`.

(Androutsopoulos et al. 1995), because the character of the data in the DB remained largely uniform over time. However, in DBs with continually evolving schemas – characteristic for the Big Data era – GUIs become untenable, because it is difficult to rapidly adapt them to the schemas.

One concrete class of examples where the DB requires non-GUI end-user interaction is websites providing semantic web annotations. By design, semantic annotations for web pages have few constraints to accommodate for the velocity and variety of content on the web (Berners-Lee et al. 2001). But because of this relative modelling freedom, querying is best left to the (rather formal) query language SPARQL. Arguably, the necessity of SPARQL may hinder the adoption of semantic web technologies by the general public.

Since GUIs can be ineffective for end-users for interacting with Big Data-era DBs, researchers have proposed alternative UIs. Perhaps one of the most intuitive ones is the natural language (NL) interface, in which users ask questions (either by typing or by voice), which are subsequently interpreted by the DB.

The idea of NL UIs is not new: NL-based DB UIs existed in as early as the 1960s (Androutsopoulos et al. 1995). However, language-related tasks by computers, notably translation, have historically proven to be difficult. Nonetheless, AI researchers have recently discovered a type of model that can effectively map symbol sequences (like words) to other symbol sequences, called the 'transformer' (Vaswani et al. 2017). Because of its general design, transformer-based models have improved the state-of-the-art on multiple natural language processing (NLP) tasks (Tenney et al. 2019). Realising that associating NL questions to DB queries can be viewed as a type of transformation, DB querying via NL UIs has attracted renewed attention.

Indeed, since the discovery of transformer models, multiple researchers have proposed NL interfaces for DBs, including those for querying various semantic web knowledge bases (see Cui et al., 2021, p. 1, for some references). One striking issue with these interfaces, however, is that they are predominantly engineered for the English language (Joshi et al. 2020). For users who have not mastered English, or prefer to ask their questions in their native language, such NL UIs thus are limiting. An example of such a language is Dutch. There has been no work that uses a monolingual Dutch transformer-model to train a NL interface to a DB, although it should be noted that the recent multilingual approach of Zhou et al. (2021) (see Section 2) includes results for Dutch as well.

Considering the opportunities that a natural language interface brings to accessing the contents of databases, our aim is to create a transformer model that is able to map Dutch NL questions to equivalent SPARQL queries for querying Wikidata (Vrandečić and Krötzsch 2014), a well-known database for semantic data. While closely following the methodology of Cui et al. (2021), we de-emphasise compositional performance of the system, and instead concentrate more on the development of a semantic parser for Dutch. To be explicit, our research question is: "Can we develop a Wikidata- and transformer-based Dutch-to-SPARQL QA system that approximates Cui et al. (2021)'s models for Hebrew, Kannada, and Mandarin Chinese in performance?" We will specify what 'performance' means in section 3.

This article is structured as follows. In section 2, we discuss QA models closely related to the one we will be developing, and specify how our system will differ from its predecessors. Then, in section 3, we tread into details concerning the design of the model, the data that will be supplied to it, as well as the model's evaluation. What follows are the results of this study, covered in section 4, which are subsequently discussed in section 5. Finally, we close with the conclusion in section 6.

2. Related Work

Because the field of question-answering over linked data has gained significant attention in the last couple of years, providing an exhaustive list of related work would be unrealistic. Instead, we refer to a set of central papers; these contain excellent pointers for further study. For a recent overview of the field, see Dimitrakis et al. (2020).

The source that primarily motivates this article is the work conducted by Cui et al. (2021). In it, the authors emphasise the need for multilingual QA systems over knowledge bases, because many state-of-the-art QA systems either work solely on English queries, or they operate on discontinued knowledge bases (notably Google's Freebase). Subsequently, Cui et al. specify a method for migrating knowledge encoded in Freebase to Wikidata, which is currently in operation, and propose and test a dataset called Compositional Wikidata Questions (CWQ), mapping natural language questions in four linguistically diverse languages (English, Hebrew, Kannada[2] and Chinese[3]). We largely follow the procedure of Cui et al., as will be shown in the methodology.

It is important to note that besides the work of Cui et al., many related and relevant QA-over-linked-data systems exist. Perhaps two of the most notable ones are WDAqua-Core 1 by Diefenbach et al. (2020) and gAnswer by Zou et al. (2014). Similar to the current proposal's system, these two models work on multilingual natural-language-to-SPARQL-queries datasets. Different from our proposed end-to-end approach, however, is that these systems both have a sequence of clearly distinct processing steps. Specifically, WDAqua-Core 1 steps through four stages: question expansion, query construction, query ranking, and finally answer decision; gAnswer, instead, has two stages: question understanding and question evaluation. More systems besides WDAqua-Core 1 and gAnswer can be found in Dimitrakis et al. (2020), p. 248. Almost without exception, these systems are not trained end-to-end. Additionally, they remain limited in their multilinguality, often only covering multiple 'high resource' languages, such as English, French, German and Spanish.

Recently, Zhou et al. (2021) have propsoed a zero-shot approach to multilingual QA. As in the experiments below, they use multilingual BERT to train a system that maps NL to SPARQL, using English as training language only. During inference, they evaluate on test questions automatically translated from English into one of several languages, including Dutch. The obtain a high F1 score for Dutch, but it should be noted that they use a predecessor of the dataset that we use for evaluation, namely LC-QuAD instead of LC-QuAD 2.0. (The latter will be discussed in Section 3.1.1.) The main difference is that LC-QuAD contains SPARQL queries for DBpedia, whereas we use Wikidata as knowledge graph.

Lastly, we stress that scholars conduct considerable research on QA systems that are very similar in character to the one proposed by us, but differ only in one or more key aspects. Following the classification in Dimitrakis et al. (2020), our proposed system is an open-domain QA system (in contrast to dialogue systems) that acts on structured data. Many systems deviate from this paradigm of question-answering. For instance, much work is done on systems that act on unstructured data, like Wikipedia excerpts. In these related paradigms, researchers are attempting to create non-English or multilingual QA systems as well. For instance, d'Hoffschmidt et al. (2020) propose FrQuAD, a document-based QA system for French.

3. Methodology

We divide our overall methodology into three sections. We begin by discussing the dataset used in this study, followed by the transformer architecture we employed. Lastly, we cover the evaluation of the transformer on the NL-to-SPARQL querying task for Dutch.

3.1 Data

3.1.1 Raw dataset

As we aim our system to be an end-to-end NL-to-SPARQL converter, our dataset's inputs should consist of natural language questions in Dutch, while our outputs should be SPARQL queries. For our study, creating such a dataset from scratch would be counterproductive for two reasons: it

2. Kannada is a Dravidic language spoken in the south of India.
3. Although not specified by Cui et al., the authors likely refer to Mandarin Chinese.

ID	Question
212,555	Which spouse of *Leonhard-Gymnasium Aachen*'s employee's sibling was influenced by *Ludwig von Beethoven*?
212,524	Which spouse of *Jin Yunying*'s female sibling was *Pujie*'s female sibling's Chinese sibling?
4,227	Did *Rachel Bilson* marry a Canadian sibling of a Canadian sibling of *Hayden Christensen*?
3,553	Did *Benjamin Keough*'s parent marry *Michael Lockwood*, *Danny Keough*, *Michael Jackson*, and *Nicolas Cage*?

Table 1: Examples of low-quality English questions in the Compositional Wikidata Questions (CWQ) dataset of Cui et al. (2021). Words in italic indicate bracketed (linked) entities in the dataset.

would be costly (both in terms of time and money), and it would make it more difficult to compare our system's performance to that of Cui et al., because the datasets would be incompatible. For this reason, we decided to adapt Cui et al.'s dataset – Compositional Wikidata Questions (CWQ) – and add Dutch variants of questions for each question-query pair in said dataset. Unfortunately, we found that many of the English questions in CWQ were of low quality, for which we found two – often related – causes: (1) sentences were phrased in an unnatural manner, and (2) an excessive amount of sentences dealt with familial relations or entertainment, which could hamper the generality of the overall system. Some problematic sentences are displayed in Table 1.

Because of the problems with CWQ, we have instead decided to adapt a different dataset for our task: the Large-scale Complex Question Answering Dataset 2.0 (LC-QuAD 2.0; Dubey et al., 2019). Although similar in input and output to CWQ, LC-QuAD 2.0 phrases questions in a more human-like manner, and covers questions outside the topic of familial relations or entertainment, such as science, geography, and more. We note that, unlike CWQ, LC-QuAD 2.0 provides queries for each question in both Wikidata and DBPedia; we used queries for the former knowledge base.

3.1.2 Processing of the raw dataset

In order to obtain a dataset that maps Dutch natural language questions to corresponding SPARQL queries from the raw LC-QuAD 2.0 dataset, we used the following processing pipeline. For each LC-QuAD 2.0 question-query pair, we followed these steps:

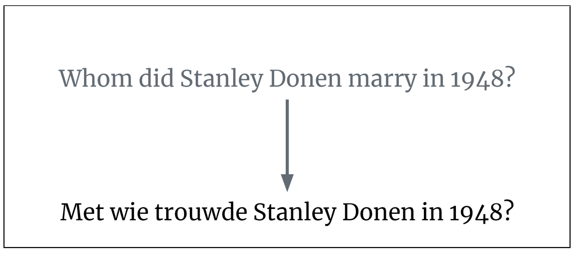
1. Translate the question from English into Dutch using Google Cloud Translate.[4]

2. Collect the entities and properties (Q- and P-values respectively, collectively named 'symbols' in this article) in the SPARQL query.

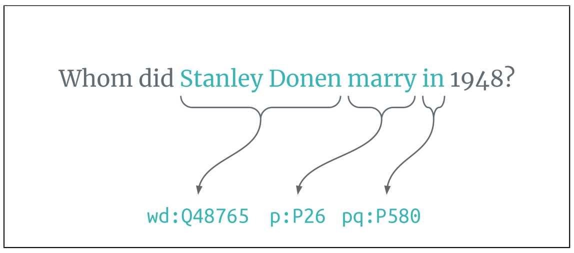
3. Find the Dutch label(s) for each of the symbols found in (2) by using the Wikidata querying service.[5]

4. Perform entity and property masking on the Dutch question. This means finding, for each symbol, that label that occurs for it in the Dutch question, and replacing these occurrences by 'masks'. For example, if the Dutch question is: "Met wie trouwde Stanley Donen in 1948?", then collected symbols in the associated query may be Q48765, P26, and P580; appropriate labels may then be 'Stanley Donen', 'trouwde', and 'in' (a Dutch temporal preposition). These three labels occurring in the question are then replaced by masks. We do this by replacing unique entities and properties by ascending pseudo-Q- and P-values: Q0, Q1, . . . , and P0, P1, ... In our example, we would replace 'Stanley Donen' by Q0, 'trouwde' by P0, and 'in' by P1. This would result in the masked Dutch question "Met wie P0 Q0 P1 1948?"

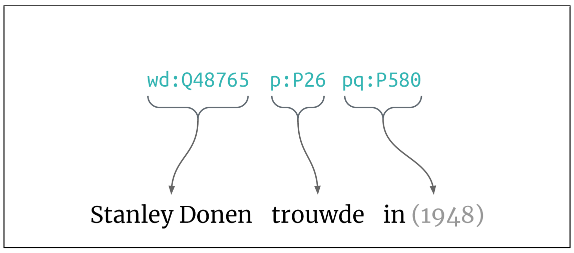
4. Information on this Google Cloud service can be found at: `https://cloud.google.com/translate/`.
5. This service can be found at `https://query.wikidata.org/`.


Whom did Stanley Donen marry in 1948?
Met wie trouwde Stanley Donen in 1948?

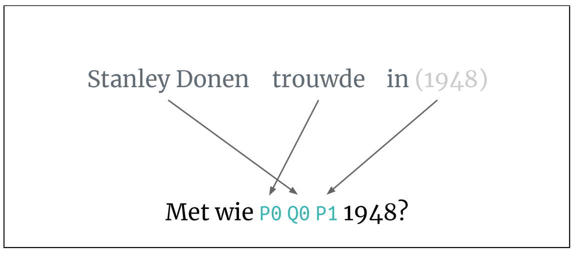
Step 1. Translate the question from English into Dutch using Google Cloud Translate.



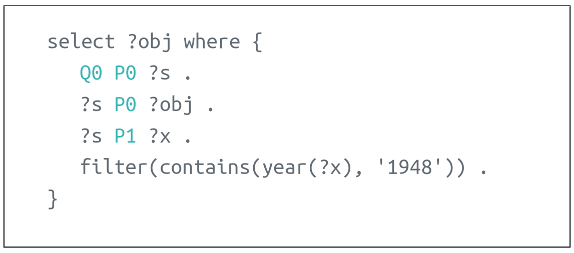
Step 2. Extract entities and properties from the SPARQL query. (Note: This figure shows how these symbols relate to the natural language question, and not the SPARQL query.)



Step 3. Find, for each symbol, the Dutch label(s) using WikiData.

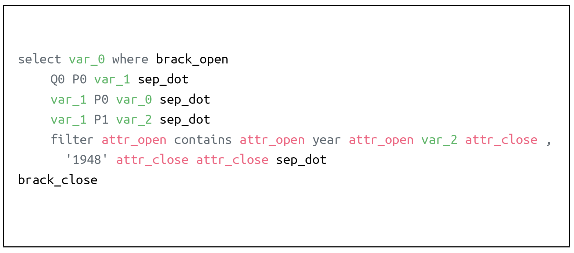


Step 4. Perform entity and property masking on the question.



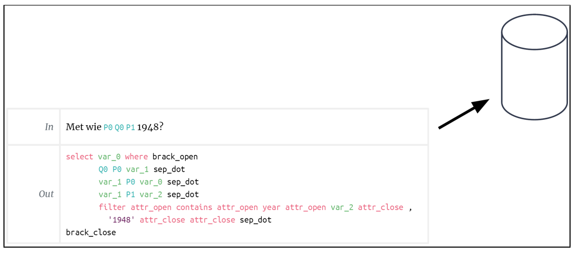
```
select ?obj where {
   Q0 P0 ?s .
   ?s P0 ?obj .
   ?s P1 ?x .
   filter(contains(year(?x), '1948')) .
}
```

Step 5. Also perform entity and property masking on the query.



```
select var_0 where brack_open
    Q0 P0 var_1 sep_dot
    var_1 P0 var_0 sep_dot
    var_1 P1 var_2 sep_dot
    filter attr_open contains attr_open year attr_open var_2 attr_close ,
      '1948' attr_close attr_close sep_dot
brack_close
```

Step 6. Post-process the masked query.



In: Met wie P0 Q0 P1 1948?

Out:

```
select var_0 where brack_open
    Q0 P0 var_1 sep_dot
    var_1 P0 var_0 sep_dot
    var_1 P1 var_2 sep_dot
    filter attr_open contains attr_open year attr_open var_2 attr_close ,
      '1948' attr_close attr_close sep_dot
brack_close
```

Step 7. Add the question-query pair to the processed set of pairs, forming our dataset.

Figure 1: A visualisation of our processing pipeline. In this example, we demonstrate how the English natural language question “Whom did Stanley Donen marry in 1948?” with accompanying SPARQL query would be processed into a datapoint for our Dutch-to-SPARQL dataset. See Section 3.1.2 for further details.

We perform this masking step for two reasons: (1) it keeps our procedure as closely related to that of Cui et al. as possible, and (2) it avoids that the model has to learn, at least in principle, all legal symbols in Wikidata, which is an unrealistic goal to achieve.

At this stage, two situations may cause us to skip a question-query pair, and to continue immediately to the next pair:

(a) When one or more symbols has no label in the Dutch question.

(b) When two or more symbol labels found in the Dutch question overlap with one another. For instance, in the Dutch question "Is de maand augustus vernoemd naar keizer Augustus?" it may be that symbol labels for 'augustus' and 'Augustus' match for one another; this would cause a conflict of this type.

Note, finally, that we use exact matching of labels against question occurrences. Thus, we require labels to exactly match lexemes in the Dutch question, and do not allow for slight deviations in spelling. Our reason for this choice is to keep the overall processing pipeline simple.

5. Perform entity and property masking on the SPARQL query. Specifically, we replace any symbol of the regular expression form `[a-z]+(:)[PQ][0-9]+` (such as `wdt:P31` or `wd:Q60`) by the corresponding masks produced in step (4).

6. Make all symbols in the resultant masked query lower-case, and replace special symbols according to the following table:

Special symbol(s)	Replacement(s)
`{`, `}`	`brack_open`, `brack_close`
`(`, `)`	`attr_open`, `attr_close`
`.`	`sep_dot`
Variables, e.g. `?ans`	`var_`*i*

Here, the i means that unique variables encountered in the original query are replaced by `var_1`, `var_2`, ...

7. Add the question-query pair to the processed set of pairs, forming our dataset.

Figure 1 visualises this process.

Following this procedure, we obtain a dataset with 2,648 question-query pairs. This is around 9% of the data before entities and properties were replaced by masks (which consists of 30,226 pairs), illustrating that most Dutch questions do not contain explicit occurrences of labels for one or more symbols in the associated queries.

3.1.3 Partitioning the dataset

Subsequently, we split the dataset into three parts: a training, validation (or development) and testing section. Since LC-QuAD 2.0 already is divided into training and testing sections, we simply follow this separation, producing a training set of 2,262 points and a testing set of 386 points. Thereafter, we divide the training set into two subsections: a (proper) training set, and a validation set. 11.1% of the data is pseudo-randomly uniformly chosen and sent to the validation set, while the remaining data is put into the training set. As such, we have 2,011 data points in the training set, 251 points in the validation set, and 386 points in the test set.

3.2 Model

Because we rely on the transformer architecture introduced by Vaswani et al. (2017), as well as BERT and its multilingual counterpart, multilingual BERT[6] (Devlin et al. 2018), we spend some space in the upcoming sections to briefly outline what these models do. Following this, we give an outline of how we combine the original transformer and the (pre-trained, multilingual) BERT model to fine-tune a complete transformer for our NL-to-SPARQL task.

3.2.1 The Transformer

In 2017, Vaswani et al. introduced the transformer architecture. The transformer is a so-called sequence-to-sequence model that, as the name implies, maps sequences of symbols to different sequences of symbols. At its introduction, the transformer was novel because it did not rely on recurrence or convolution—techniques that before were considered important for sequence transformation. Instead, the transformer relies on attention. More specifically, the model uses multiple, stacked attention layers (called 'heads') that enable it to jointly attend to multiple, distinct parts of the input representation. Combined with positional encodings—specially chosen values added to embedding versions of the input sequence's tokens to encode position—the transformer's attention layers can effectively work with sequential data. Further, it is important to note that the transformer consists of two parts: an encoder that derives numerical representations for input symbolic sequences, and a decoder that, from these representations, yields outputs from which another (transformer) symbol sequence can be built. For more detailed information on the transformer, we refer the interested reader to Vaswani et al. (2017).

3.2.2 BERT

BERT (Bi-directional Encoder Representations from Transformers; Devlin et al. 2018) is a language model that is nearly identical to the encoder part of the original transformer architecture. Arguably, the main innovation made by BERT is its general applicability across many NLP tasks as a pre-trained model, requiring few to no changes to the model itself in many cases. This wide applicability is enabled by the input representation, which allows users to submit one or two sentences in one sequence, plus the way it has been pre-trained. By adding one or more neural layers at the final layer of BERT and fine-tuning this complete model, users can achieve remarkable results in multiple natural language processing tasks. As with the transformer architecture, more detailed information on BERT can be found in Devlin et al. (2018).

We further note that a multilingual version of BERT ('multilingual BERT') has been released since 3 November 2018;[7] it has been trained on the top-100 most populous Wikipedias, as determined by their numbers of articles. Dutch is part of this list.

3.2.3 BERT-based Transformer model

Since BERT is strongly related to the transformer architecture, an intuitive idea is to incorporate BERT back into the transformer architecture to obtain a 'BERT-based' transformer model that has been pre-trained. Fine-tuning this transformer would then lead to a full transformer model that can be used in sequence-to-sequence NLP tasks. Although we can use BERT as an encoder without much further adaptation to the transformer, using BERT at the decoder-side is more involved, because the decoder is not simply an inversely-layered encoder. We instead build on the approach of Tran et al. (2021, p. 6) and make the following two changes: (1) to convert the attention sub-layers in the decoder from bi-directional to leftward-only, we use a masking mechanism as suggested in Vaswani et al. (2017), and (2) to take into account the additional cross-attention sub-layer (connecting this

6. Other names for this model are 'mBERT' or 'ML-BERT'.
7. According to the information provided in the `README.md` of `https://github.com/google-research/bert`—the repository of the BERT model.

decoder layer to a corresponding encoder layer), we initialise the cross-attention weights randomly from a $\mathcal{U}(-\sqrt{1\,/\,n_{\text{in}}}, \sqrt{1\,/\,n_{\text{in}}})$ distribution, where $n_{\text{in}} \in \mathbb{N}^+$ is the number of features of a single input entry to said cross-attention.

For our Dutch-to-SPARQL model, we start with the pre-trained base and cased version of multilingual BERT. Architecturally, this means that the transformer's encoder and decoder BERT models both consist of $A = 12$ self-attention heads, $H = 768$ entries per hidden layer, and $L = 12$ of such layers, and that distinctions are made between upper- and lowercase letters in the symbols on the input and output.

Regarding our implementation, we have adapted the codebase provided by Tran et al. (2021) to obtain an implementation for our Dutch-to-SPARQL model. As such, we use weight tying on the input- and output embeddings (p. 6) as well as beam search (instead of greedy search) on selection of output tokens, with a beam size of $k = 10$. Besides this, we limit the NL input length to 64 tokens and the SPARQL output length to 128 tokens.

3.3 Training and Evaluation

3.3.1 Training

Recall from Section 3.2.3 that our model design begins with an en- and decoder that have both already been pre-trained, albeit separately. 'Training' in our methodology, then, actually means that we fine-tune the transformer, departing from a parameterisation that is likely better than a random initialisation of weights. We still call the fine-tuning operation 'training', because the model in its entirety is re-parameterised to function well on the specific task of Dutch-to-SPARQL translation.

In order to train our BERT-based transformer model, we used the Adam optimiser (Kingma and Ba 2014) with a default learning rate of $1 \cdot 10^{-5}$, and with $\beta_1 = 9 \cdot 10^{-1}$, $\beta_2 = 9.99 \cdot 10^{-1}$, and $\epsilon = 1 \cdot 10^{-6}$.

Adam attempts to minimise the label-smoothed cross-entropy loss. The label-smoothing in this loss serves to reduce the model's certainty in its token predictions, thereby reducing its propensity to overfit to training samples. This label-smoothed cross-entropy loss, then, is defined as follows. Let $B \in \mathbb{N}^+$ be the number of samples per batch, $L_{\text{out}} \in \mathbb{N}^+$ be the maximum output sequence length in tokens, and $V_{\text{out}} \in \mathbb{N}^+$ be the output vocabulary size. Then, let $X \in \mathbb{R}^{B \times L_{\text{out}} \times V_{\text{out}}}$ be the decoder's output, used to compute predicted tokens per sequence, and let $Y \in \{0,1\}^{B \times L_{\text{out}} \times V_{\text{out}}}$ be embedded, ground-truth token sequences. Note here that each vector $Y_{b,i,:}$ is one-hot coded over the output vocabulary (where $1 \leq b \leq B$ and $1 \leq i \leq L_{\text{out}}$). With these symbols defined, the label-smoothed cross-entropy loss, denoted by $\mathcal{L}_{\text{LSCE}}(X, Y)$, is given by

$$\mathcal{L}_{\text{LSCE}}(X,Y) = -\frac{1}{B \times L_{\text{out}}} \sum_{b=1}^{B} \sum_{i=1}^{L_{\text{out}}} \text{LabelSmooth}_{\ell}(Y_{b,i,:}) \cdot \ln\left(\frac{X_{b,i,:}}{\sum_{j=1}^{V_{\text{out}}} X_{b,i,j}} \right), \tag{1}$$

where $\ell \in [0, 1]$ is the so-called label smoothing scalar that 'smoothes out' or 'redistributes' the probability in the one-hot vectors $Y_{b,i,:}$ as follows:

$$\text{LabelSmooth}_{\ell}(Y_{b,i,j}) = \begin{cases} 1 - \ell & \text{if } Y_{b,i,j} = 1, \\ \ell\,/\,(V_{\text{out}} - 1) & \text{if } Y_{b,i,j} \neq 1, \end{cases} \tag{2}$$

$$\text{for all } j \in [1, V_{\text{out}}]. \tag{3}$$

In all our experiments, we set ℓ equal to 0.1.

Further, we add a weight decay (L_2 regularisation term) scaled by $1 \cdot 10^{-2}$ to the optimiser, where we apply the decay to all transformer parameters except the biases and layer normalisation weights (Ba et al. 2016). Besides this, we set the default batch size to 32 samples, and make the training stage consist of 200 epochs.

#	Hyper-parameter	Change (default $\rightarrow$ alternative)
1	–	–
2	Batch size	$32 \rightarrow 8$
3	Batch size	$32 \rightarrow 16$
4	Learning rate	$1 \cdot 10^{-5} \rightarrow 5 \cdot 10^{-6}$
5	Learning rate	$1 \cdot 10^{-5} \rightarrow 2 \cdot 10^{-5}$
6	Learning rate	$1 \cdot 10^{-5} \rightarrow 3 \cdot 10^{-5}$
7	BERT model	Multilingual BERT $\rightarrow$ BERT

Table 2: The experimental conditions tested in this study in order to find the appropriate hyper-parameters for our transformer. Experimental condition #1 is the base condition against which all other conditions are compared.

3.3.2 Experiments

We conduct two distinct sets of experiments in this study. The base experiments directly serve to answer our research question, whereas the additional experiments consider in more detail what the effect is of varying the transformer's en- and decoder on performance on the Dutch-to-SPARQL translation task.

Base experiments. In order to discover what hyper-parameters are appropriate for our transformer, we vary the batch size, learning rate and type of pre-trained en- and decoder BERT model. We begin by considering a base hyper-parameterisation: a batch size of 32, a learning rate of $1 \cdot 10^{-5}$, and a multilingual BERT model. This base hyper-parameterisation corresponds to condition #1 in Table 2. We then measure the validation corpus-level BLEU score after training for 200 epochs. Following this measurement, in six conditions, we change the value of a single hyper-parameter and re-measure the validation BLEU score. These value-changes correspond to conditions #2 up until #7 in Table 2. For instance, in two of these conditions we change the value of the batch size hyper-parameter, namely from its base value of 32 to sizes of 8 and 16 (conditions #2 and #3 in Table 2, respectively). Subsequently, for each hyper-parameter, we determine which setting produced the highest validation corpus-level BLEU score—the default value or one of its alternatives—and use that value for our final transformer configuration. Lastly, we train this final transformer configuration for 200 epochs and evaluate it by computing its BLEU score on the test split of our dataset.

Note that our procedure is not a complete grid search over the three hyper-parameters. The grid-search approach is to be preferred, because it takes into account potential interactions between hyper-parameter value-changes. However, we do not opt for said approach because of its computational demands.

Additional experiments. Given the final transformer configuration from the base experiments, we conduct six additional experiments by varying the en- and decoder of the transformer; in a sense, this is an expansion of the variation in the 'BERT model' hyper-parameter of the base-experiments. The six experiments correspond to six encoder-decoder combinations. In these, we involve two new encoders and two new decoders, different from the base experiments' multilingual BERT:

#	Encoder	Decoder	#	Encoder	Decoder
1	BERTje	Multilingual BERT	4	BERTje	XLM-RoBERTa
2	Multilingual BERT	SPBERT	5	XLM-RoBERT	SPBERT
3	BERTje	SPBERT	6	XLM-RoBERTa	XLM-RoBERTa

Here, BERTje (de Vries et al. 2019) and SPBERT (Tran et al. 2021) act as a novel encoder and decoder that have been pre-trained specifically on Dutch texts and SPARQL query logs, respectively; XLM-

#	Hyper-parameter	Change (old → new)	BLEU
1	–	–	51.72
2	Batch size	$32 \rightarrow 8$	52.54
3	Batch size	$32 \rightarrow 16$	52.00
4	Learning rate	$1 \cdot 10^{-5} \rightarrow 5 \cdot 10^{-6}$	49.54
5	Learning rate	$1 \cdot 10^{-5} \rightarrow 2 \cdot 10^{-5}$	52.59
6	Learning rate	$1 \cdot 10^{-5} \rightarrow 3 \cdot 10^{-5}$	49.96
7	BERT model	Multilingual BERT → BERT	50.32

Table 3: Final validation BLEU scores for each condition from the base experiments. For each experimental condition listed in Table 2, the corpus-level BLEU score of the trained system after 200 epochs is shown.

RoBERTa (Conneau et al. 2019) serves as both an encoder and a decoder, much like multilingual BERT in the base experiments. All these models follow the original BERT's base-and-cased design, as explained in Section 3.2.3.

Once again, we record the final training epoch's validation corpus-level BLEU score to determine the relative effectiveness of the resultant transformer configuration.

3.3.3 Evaluation

We evaluate transformer configurations using the corpus-level Bi-Lingual Evaluation Understudy (BLEU) measure (Papineni et al. 2002)—a standard within sequence-to-sequence tasks. Note that BLEU scores are often reported as a value within $[0, 100]$ instead of one within $[0, 1]$, achieved by scaling the original fraction by 100. We follow this custom.

4. Results

The results section is divided into three parts, which form two subsections. In the first part, we (I) present the base experiments' corpus-level BLEU scores per each experimental condition, and (II) conclude which transformer configuration is the 'best' in the greedy sense. Second, we list, per each encoder-decoder combination from the additional experiments, what its final corpus-level validation stage BLEU score is given the greedy transformer configuration. Collectively, we call these first two parts the 'quantitative' subsection of the results. Then, third and last, we move on to a qualitative inspection of predicted queries made by the greedily-selected 'best' transformer, where we investigate what the model does well and where it fails, and how.

4.1 Quantitative results

4.1.1 Base experiments

The quantitative results are shown in Table 3. We observe that decreasing the batch size from 32 to either 16 or 8 appears to slightly improve the BLEU score of the system, although the change is very minimal. For the learning rate, we see that slightly raising the rate—from $1 \cdot 10^{-5}$ to $2 \cdot 10^{-5}$— appears to raise the transformer's BLEU score lightly; strongly decreasing the rate (to $5 \cdot 10^{-6}$) or raising the rate above $2 \cdot 10^{-5}$ causes the system's performance to drop by around 2 BLEU. Finally, If we replace the multilingual base-and-cased BERT model by the English-only base-and-cased BERT model, validation set performance slightly decreases as well.

#	Encoder	Decoder	BLEU
1	BERTje	Multilingual BERT	52.31
2	Multilingual BERT	SPBERT	52.60
3	BERTje	SPBERT	54.75
4	BERTJE	XLM-RoBERTa	0.0
5	XLM-RoBERTa	SPBERT	54.29
6	XLM-RoBERTa	XLM-RoBERTa	18.20

Table 4: Final validation BLEU scores for each of the six additional experimental configurations, in which we change the selection of encoder and decoder. As is the case in Table 3, scores are corpus-level BLEU scores after training for 200 epochs, and evaluating on the validation set.

Given these results, we greedily selected the best hyper-parameter values according to Table 3—a batch size of 8, a learning rate of $2 \cdot 10^{-5}$, and using multilingual BERT for the encoder and decoder of the transformer. This transformer was then trained similarly as the other hyper-parametrised models (final validation BLEU score: 49.57), and was subsequently tested on the test partition of our Dutch-to-SPARQL dataset. The BLEU score on this test set was 51.86.

4.1.2 Additional experiments

Table 4 displays the validation corpus-level BLEU scores we obtained for each transformer configuration. Two observations can be made here. First, it appears that the transformers produce validation BLEU scores that are greater than the final validation BLEU score obtained by the original greedily-selected model configuration from the main experiments, apart from configurations 4 and 6. Second, configurations 4 and 6 produce markedly lower validation BLEU scores than any of the original or extra experimental settings.

4.2 Qualitative results

In addition to collecting the quantitative results, presented above, we have inspected predictions made on the test partition by our greedily-selected 'best' transformer model from the base experiments. Here, we found some patterns that are of interest in explaining the obtained BLEU score:

1. When the ground-truth query uses a `COUNT` statement to count the number of bound variables, our transformer instead chooses to use `SELECT DISTINCT`. As a result, it obtains the same answer as the ground truth query, but it does something different with it: it reports the answers as-is instead of counting them. The inverse also happens occasionally: the prediction is a `COUNT` query, while the ground-truth query is one involving the `DISTINCT` statement.

2. Some queries involve matching primitive values. For instance, multiple queries try to obtain locations by providing coordinates, requiring models to formulate statements of the form

   ```
   ?v wdt:PP625 ?w FILTER(CONTAINS(?w, 'lat.long'))
   ```

   where `lat` and `long` are the latitude and longitude of a location on earth in degrees. What we notice is that, while the ground-truth sequence correctly separates tokens in such queries, our model incorrectly joins multiple of these tokens together. For example, it may predict `"var_2,138"` instead of `"var_2", ",", "138"`. Understandably, this hurts the BLEU score, from unigrams to 4-grams.

3. The model sometimes swaps entities or properties across statements, or, when within a statement, the statement's head- and tail entities. This seems more likely to happen when the expected query is relatively long.

4. Although this occurs more sporadically, it sometimes happens that the model predicts a semantically identical SPARQL query to the ground-truth query, but where it names the variables differently. For example, the following two queries are semantically identical, but not literally identical:

   ```
   SELECT ?v1 WHERE { ?v1 P1 Q1 }
   ```
   versus
   ```
   SELECT ?v2 WHERE { ?v2 P1 Q1 }
   ```

   This strongly impacts the BLEU score as well, as n-grams, for all $n \in [1, 4]$, are affected.

5. Overall, we notice that longer queries have a higher probability of mismatching with the ground-truth query. The inverse applies to relatively short queries.

5. Discussion

5.1 Comparison to Cui et al.

In the introduction, we posed ourselves the question whether it is possible to create a transformer-based model that is capable of reaching or surpassing the performance of the non-English models in Cui et al.. In that paper, BLEU scores on the test set were 66.0 for Hebrew, 45.6 for Kannada, and 58.6 for (Mandarin) Chinese.[8] If we compare those scores to the final BLEU value of our tested transformer (51.86), then it appears that our model surpasses the Kannada model of Cui et al., but remains inferior in performance to the Hebrew and Mandarin Chinese models.

We propose two reasons for the difference in BLEU score, as compared to Cui et al.. First, note that a direct comparison between Cui et al.'s results and that of our model is not fully valid, as we use different datasets. In particular, keep in mind that Cui et al.'s dataset has more than twice the amount of data points for training and validation: 4,354 versus 2,011 for training and 585 versus 251 for validation, and around 50% more data points for testing (571 such points versus 386 in our dataset). Especially for deep learning systems, even when they have been pre-trained, more data generally results in better performance.

Second, recall that we chose not to use Cui et al.'s CWQ dataset in part because the problem domain was limited: it mainly dealt with familial relations and entertainment. We avoided this by constructing our own dataset based on LC-QuAD 2.0, a dataset that covers more general topics. However, because of its greater generality, it may be that our BLEU score is harmed in the process, because the model needs to learn a more general mapping from natural language to SPARQL.

5.2 Performance on the dataset

Besides the comparison with Cui et al., it may be worthwile to consider what may inhibit greater performance of the transformer on our dataset, considered apart from performances of other authors' models. The qualitative results from Section 4.2 suggest that multiple factors may be at play in preventing better BLEU scores. One may be ambiguity: variable namings (point 4) and ordering of statements in a `SELECT` block can be changed without losing semantic accuracy. Given that there is such variability in LC-QuAD 2.0 queries, ambiguity can be realistically attributed a role in lowered BLEU scores. Another factor may be (relatively) rare constructs, such as the use of `DISTINCT` and `FILTER` commands (point 1). If not enough examples of such queries exist in the dataset—and our dataset is relatively small—then it is probable that the model makes incomplete or incorrect predictions when these terms are required in the ground-truth queries.

8. We work with the random test split of that paper, see p. 9. Cui et al. also provide performances on specially-designed splits, called Maximum Compositional Divergence (MCD) splits. Since we use an essentially random partitioning scheme however, we do not work with said divergence splits in this article.

Although it is not applicable to our experiments, it is important to realise that the transformer cannot rely on masking when it is used 'in production'. We expect the effect on predicted queries to be limited because the masking is designed to make predictions robust to variation in nouns and verbs—roughly, the natural language counterparts to Q- and P-values. However, we have not investigated this topic.

5.3 Findings from the additional experiments

During the second part of the Results (Section 4.1.2), we made two observations. First, except for configurations 4 and 6, the additional experiments' transformers produce validation BLEU scores that appear to surpass the final validation BLEU found when training the final, greedily-selected 'best' transformer. Second, configurations 4 and 6 yield notably lower validation BLEU scores than the other transformer settings.

The first observation may be explained by the fact that all settings *except* configurations 4 and 6 replace their encoder or decoder (or both) by language models that are arguably better suited to the task of translating Dutch questions into corresponding SPARQL queries, in two distinct senses: (I) The language model that replaces the original multilingual BERT is pre-trained specifically for the natural- or query language at said side of the transformer. For example, the SPBERT decoder is pre-trained on SPARQL query logs, which may be beneficial in decoding to SPARQL query output sentences. (II) The pre-trained model attained better results on the pre-training tasks, in comparison to the original multilingual BERT model. XLM-RoBERTa, for instance, sees two-digit percentage gains when applied on tasks such as XNLI, MLQA, and others (Conneau et al. 2019). As such, said model may be more promising for downstream tasks as well.

Observation two—that the extra experimental configurations 4 and 6 appear to obtain lower validation corpus-level BLEU scores than the other settings—is perhaps less easily understood. The pattern shared between the two configurations is that they both use XLM-RoBERTa as the decoder language model—something that does not apply to any other experimental setting. Thus, there may be a problem between XLM-RoBERTa and the to-be-predicted queries, such as the tokeniser of the former not having certain crucial symbols in its vocabulary for formulating SPARQL expressions. Still, the difference in BLEU is arguably too great for this to be the (only) reason.

6. Conclusion

Our research question that we have sought to answer is: "Can we develop a Wikidata- and transformer-based Dutch-to-SPARQL QA system that approximates Cui et al.'s models for Hebrew, Kannada, and Mandarin Chinese in performance?" By incorporating a multilingual BERT model in the encoder and decoder components of the original transformer architecture of Vaswani et al. (2017), and by training and validating this model on a specially designed Dutch-to-SPARQL dataset derived from LC-QuAD 2.0 (Dubey et al. 2019), we proposed exactly such a system. Our quantitative analysis suggested that, although the base experiments' finally-selected transformer model does not surpass Cui et al.'s Hebrew and Mandarin Chinese model, we do improve over the Kannada model. As such, we may regard our system as having partially confirmed the research question. However, as we have noted in the results, a direct comparison between Cui et al. and our results is not completely possible, as we use different datasets; as such, a comparison of BLEU scores may be misleading.

The qualitative analysis of our transformer suggests multiple points where our study may be problematic: query ambiguity and, simply, a deficiency of data likely have reduced our BLEU score, and do not make our system ready-for-use in any actual application domains; further investigation into this model's functioning will be needed to get to such a point.

This article may be an interesting starting point for future research. As we have also hinted at in the discussion section of our results, it may be fruitful to improve on the Dutch-to-SPARQL dataset that we have built for this study, overcoming the ambiguities we noted in the qualitative analysis, and expanding on its size so that it can be competitive with CWQ or LC-QuAD 2.0. Moreover, it may be worthwhile to explore how variations in the encoder-decoder combination—explored in the additional experiments—interact with changes in the base experiments' hyper-parameters; it may be that further improvements in testing partition BLEU may be obtained in this way.

All in all, we hope that this work helps to raise awareness of and interest in the importance of linguistic diversity in our natural language processing models, thereby getting closer to developing information systems that are accessible to all.

References

Androutsopoulos, Ion, Graeme D. Ritchie, and Peter Thanisch (1995), Natural Language Interfaces to Databases – An Introduction, *Natural Language Engineering* **1** (1), pp. 29–81, Cambridge University Press. DOI: 10.1017/S135132490000005X.

Ba, Jimmy Lei, Jamie Ryan Kiros, and Geoffrey E. Hinton (2016), Layer Normalization. arXiv pre-print. Identifier: 1607.06450.

Berners-Lee, Tim, James Hendler, and Ora Lassila (2001), The Semantic Web, *Scientific American* **284** (5), pp. 34–43, Springer Nature America. URL: `https://www.jstor.org/stable/26059207`.

Conneau, Alexis, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov (2019), Unsupervised Cross-Lingual Representation Learning at Scale. arXiv pre-print. Identifier: 1911.02116.

Cui, Ruixiang, Rahul Aralikatte, Heather Lent, and Daniel Hershcovich (2021), Multilingual Compositional Wikidata Questions. arXiv pre-print. Identifier: 2108.03509.

de Vries, Wietse, Andreas van Cranenburgh, Arianna Bisazza, Tommaso Caselli, Gertjan van Noord, and Malvina Nissim (2019), BERTje: A Dutch BERT Model. arXiv pre-print. Identifier: 1912.09582.

Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova (2018), BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv pre-print. Identifier: 1810.04805.

d'Hoffschmidt, Martin, Wacim Belblidia, Tom Brendlé, Quentin Heinrich, and Maxime Vidal (2020), FQuAD: French Question Answering Dataset. arXiv pre-print. Identifier: 2002.06071.

Diefenbach, Dennis, Andreas Both, Kamal Singh, and Pierre Maret (2020), Towards a Question Answering System over the Semantic Web, *Semantic Web* **11** (3), pp. 421–439, IOS Press. DOI: 10.3233/SW-190343.

Dimitrakis, Eleftherios, Konstantinos Sgontzos, and Yannis Tzitzikas (2020), A Survey on Question Answering Systems over Linked Data and Documents, *Journal of Intelligent Information Systems* **55**, pp. 233–259, Springer-Verlag. DOI: 10.1007/s10844-019-00584-7.

Dubey, Mohnish, Debayan Banerjee, Abdelrahman Abdelkawi, and Jens Lehmann (2019), LC-QuAD 2.0: A Large Dataset for Complex Question Answering over Wikidata and DBpedia, *in* Ghidini, Chiara, Olaf Hartig, Maria Maleshkova, Vojtěch, Svátek, Isabel Cruz, Aidan Hogan, Jie Song, cois Maxime Lefran and Fabien Gandon, editors, *ISWC '19: Proceedings of the 18th International Semantic Web Conference*, Vol. 18 of *ISWC: International Semantic Web Conference*, pp. 69–78. DOI: 10.1007/978-3-030-30796-7_5.

Jain, Anil (2016), The 5 V's of Big Data. Web page. Retrieved 25 July 2022, from `https://www.ibm.com/blogs/watson-health/the-5-vs-of-big-data/`.

Joshi, Pratik, Sebastin Santy, Amar Budhiraja, Kalika Bali, and Monojit Choudhury (2020), The State and Fate of Linguistic Diversity and Inclusion in the NLP World, *in* Jurafsky, Dan, Joyce Chai, Natalie Schluter, and Joel Tetreault, editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Vol. 58, ACL, pp. 6282–6293. DOI: 10.18653/v1/2020.acl-main.560.

Kingma, Diederik P. and Jimmy Ba (2014), Adam: A Method for Stochastic Optimization. arXiv pre-print. Identifier: 1412.6980.

Papineni, Kishore, Salim Roukos, Todd Ward, and Wei-Jing Zhu (2002), BLEU: A Method for Automatic Evaluation of Machine Translation, *in* Isabelle, Pierre, editor, *ACL '02: Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, Vol. 40, ACL, pp. 311–318. DOI: 10.3115/1073083.1073135.

Tenney, Ian, Dipanjan Das, and Ellie Pavlick (2019), BERT Rediscovers the Classical NLP Pipeline. arXiv pre-print. Identifier: 1905.05950.

Tran, Hieu, Long Phan, James Anibal, Binh T. Nguyen, and Truong-Son Nguyen (2021), SPBERT: An Efficient Pre-training BERT on SPARQL Queries for Question Answering over Knowledge Graphs. arXiv pre-print. Identifier: 2106.09997.

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, and Łukasz Kaiser (2017), Attention is All You Need. arXiv pre-print. Identifier: 1706.03762.

Vrandečić, Denny and Markus Krötzsch (2014), Wikidata: A Free Collaborative Knowledgebase, *Communications of the ACM* **57** (10), pp. 78–85, Association for Computing Machinery. DOI: 10.1145/2629489.

Zhou, Yucheng, Xiubo Geng, Tao Shen, Wenqiang Zhang, and Daxin Jiang (2021), Improving Zero-Shot Cross-lingual Transfer for Multilingual Question Answering over Knowledge Graph, *in* Hakkani-Tur, Dilek, Anna Rumshisky, and Luke Zettlemoyer, editors, *NA ACL '21: Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, ACL, pp. 5822–5834. DOI: 10.18653/v1/2021.naacl-main.465.

Zou, Lei, Ruizhe Huang, Haixun Wang, Jeffrey Xu Yu, Wenqiang He, and Dongyan Zhao (2014), Natural Language Question Answering over RDF: A Graph Data Driven Approach, *in* Dyreson, Curtis and Feifei Li, editors, *SIGMOD '14: Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, Vol. 40 of *SIGMOD: Management of Data*, pp. 313–324. DOI: 10.1145/2588555.2610525.