UTD HLTRI at TREC 2018: Complex Answer Retrieval Track

Ramon Maldonado and Sanda M. Harabagiu Department of Computer Science Human Language Technology Research Institute University of Texas at Dallas U.S.A {ramon, sanda}@hlt.utdallas.edu

Abstract

Finding answers to complex questions within a corpus of Wikipedia paragraphs needs to account for (a) the similarity between questions and paragraphs as well as (b) their shared semantics. In our participation in the 2018 TREC CAR track, we focused on developing a novel neural paragraph ranking model in our existing CAPAR system, developed for the 2017 TREC CAR track. The new system TRANS-CAPAR, takes advantage of the recently introduced Transformer architecture to encode information from the question and to semantically decode it in each paragraph. The results obtained during the official evaluations indicate that TRANS-CAPAR makes good use of both discourse context and similarity when ranking paragraphs.

1 Introduction

In our second year of participation in the Text REtrieval Conference Complex Answer Retrieval (TREC CAR) track we have focused only on the paragraph retrieval task, which enabled us to benefit from our Complex Answer PAragraph Retrieval (CAPAR) system designed for our participation in last year's TREC CAR track. Finding relevant paragraphs that answer a complex question is essential for satisfying the information need expressed by the complex question (CQ). Complex questions, as illustrated in Figure1, have the format of a title and a table of contents, with entries having a *path* using the words from the title, section and corresponding subsection. For example, the entry for subsection 2.5 in the complex question from Figure1 is represented by the path Analytics/Applications of analytics/Digital analytics. When complex questions are processed, de-

Title:

Analytics
Table of Contents:
1. Analytics vs. analysis
2. Application of analytics
2.1 Marketing optimization
2.2 People analytics
2.3 Portfolio analytics
2.4 Risk analytics
2.5 Digital analytics
2.6 Security analytics
2.7 Software analytics
3. Challenges
4. Risks

Figure 1: Example of complex question as the topic "Analytics". Each complex question is represented by (1) the title of an article and (2) its table of contents.

ciding which paragraph is relevant for each entry of the CQ's Table of Contents (TOC) needs to take into account not only the path of the TOC entry, but also additional relevance and discourse signals. We believe that discourse cohesion and coherence contributes to the relevance of a paragraph to a CQ's TOC entry. For example, coreference, as a form of discourse cohesion, holds multiple paragraph "together" in answering an entry of the TOC. Moreover, coherence relations, such as elaboration, also account for the ability to inform the answer, and thus decide on the ranking of paragraphs.

As any possible cues to inter-paragraph cohesion and coherence are removed when an enormous "bag of paragraphs" are presented for selection and ranking for each entry of a complex question's TOC, we asked ourselves if discourse



Figure 2: The Architecture of the TRANS-CAPAR system.

semantics alone could be used and thus use a more sophisticated neural pairwise ranker, operating between pairs of Wikipedia paragraphs would be capable of learning a ranking function that can account for similarity but also "pay attention" to patterns signaled from the *discourse context* of each paragraph. While many neural attention mechanisms are available, we chose to use the recently introduced Transformer architecture (Vaswani et al., 2017) as we believed that its multi-headed attention could capture the complex context patterns within paragraphs.

The remainder of the paper is organized as follows: Section 2 presents the new architecture of our system, Section 3 provides results, which are analyzed in Section 4 while Section 5 summarizes the conclusions.

2 The Approach

The paragraph retrieval task of TREC CAR requires teams to retrieve a ranked list of paragraphs for each section of an article outline representing aspects of a complex topic. To address this task, we developed the TRANSformer Complex Answer PAragraph Retrieval (TRANS-CAPAR) system having the architecture illustrated in Figure 2. TRANS-CAPAR is an extension of our complex answer retrieval system from last year, CA-PAR (Maldonado et al., 2017) that leverages the Transformer architecture (Vaswani et al., 2017) to improve paragraph ranking. The TRANS-CAPAR architecture consists of the same five modules as CAPAR:

1. Query Processing Module. Given a complex question (CQ), corresponding to an arti-

cle title, either from the training or the testing set, the article outline is provided to the Query Processing Module in order to produce a set of queries for each section of the outline, used for searching relevant paragraphs at testing time.

- Paragraph Indexing Module creates a searchable index of paragraphs from Wikipedia articles using Lucene ¹ version 5.2.1.
- 3. **Paragraph Search Module** searches a query against the paragraph index using BM25 resulting in a list of relevant paragraphs for each section in an article outline.
- 4. Feature Extraction Module extracts dynamic word- and entity-level features from each query/paragraph pair.
- 5. **Paragraph Ranking Module** produces a separate ranking of the retrieved paragraphs for each section using a novel deep neural relevance model: The Siamese Transformer Network.

Modules 1-4 operate exactly as in the CA-PAR system (Maldonado et al., 2017), however the Paragraph Ranking Module has been replaced with a novel neural relevance model that combines strategies that were shown to be successful last year with improved semantic relevance matching via the Transformer model. Therefore, we describe next only the new paragraph ranking module.

¹http://lucene.apache.org/

Paragraph Ranking

To create our new paragraph rankings, we trained The Siamese Transformer Network (STNet) for Pairwise Ranking, within a neural Learning-to-Rank (L2R) system. Given a section s and a set of paragraphs P, STNet produces a relevance score for each paragraph $p \in P$ and the final rankings are produced by sorting P by relevance score. Since the L2R system is the same as in our CA-PAR system (Maldonado et al., 2017), we detail next only the Siamese Transformer Network.

2.1 Siamese Transformer Network for Pairwise Ranking

The Siamese Transformer Network (STNet) for Pairwise Ranking is a deep neural network that learns to rank pairs of paragraphs according to their relevance to a Wikipedia article section. In order to produce a paragraph ranking, STNet learns to calculate a relevance score for a query expressed as a section heading path against paragraphs that are relevant to it.

STNet is a neural architecture using in a novel way an encoder, a decoder and an attention mechanism. Typically, encoder-decoder architectures using attention are used in sequence-to-sequence models to translate an input sequence into an output sequence. Instead of translating an input sequence into an output sequence, we cast the encoder as having the role of capturing a representation of the words used in the section heading path while the decoder generates a semantic representation of how well a paragraph matches the query. The encoder-decoder architecture is implemented as a Transformer (Vaswani et al., 2017), which uses both self-attention and traditional attention to (a) generate contextualized representations of both the query and paragraph and (b) semantically match words from the query to words from the paragraph.

In addition to the Transformer, we have used (1) a feature encoder to capture dynamic features between the query and paragraph and (2) a neural similarity module that implements PACRR (Hui et al., 2017) based on the top-performing system from the 2017 TREC CAR evaluation (MacAvaney et al., 2018). We hypothesized that combining a similarity-matrix-based approach with the sophisticated representation provided by a the Transformer Seq2seq model should improve results as the two approaches should encode complimentary information from the input text.

Given the text and features of two paragraphs along with the text of a section heading path, STNet determines which of the two paragraphs is more relevant to the section using the following five main components illustrated in Figure 3(a):

- The Transformer Encoder, detailed in Figure 3(b), encodes the sequence of words in a section heading path into a sequence of vector encodings that can be used to perform semantic soft-matching against the words of the two paragraphs;
- The Transformer Decoder, detailed in Figure 3(c), determines which pairs of paragraph and section words indicate a relevance match and produces a vector representing this softmatching;
- The Neural Similarity Module, detailed in Figure 3(d), computes a cosine similarity matrix between a section path and a paragraph, uses CNN layers to extract n-gram similarity patterns, and produces a vector representation representing the strongest similarity signals using a RNN;
- 4. **The Feature Encoder** encodes the raw features for each section-paragraph pair into a single vector encoding;
- 5. The Scorer combines the feature encoding, the similarity encoding, and the attention vector into a single real valued score $\phi(s, p)$ representing the relevance of paragraph p to section s.

STNet is a Siamese network that shares weights among the Feature Encoders, Neural Similarity Modules, Transformer Encoders and Decoders, and Scorers for both input paragraphs.

We trained the network by optimizing the pairwise hinge loss between the scores from the two halves of the Siamese network:

$$l(s, p_1, p_2) = [1 + \tau(p_1, p_2) (\phi(s, p_1) - \phi(s, p_2))]_+$$
(1)

where $[\cdot]_+$ is the hinge loss and

$$\tau(p_1, p_2) = \begin{cases} 1 & \text{if } p_1 \text{ is more relevant than } p_2 \\ -1 & \text{otherwise.} \end{cases}$$
(2)



Figure 3: The Siamese Attention Network for Pairwise Ranking. The full network is illustrated in (a), the Transformer Encoder and Decoder are depicted in (b) and (c) respectively, the Neural Similarity module is shown in (d) and the Feature Encoder is shown in (e).

STNet was trained on the train-v2.0 dataset, using the test200.v2.0 dataset for validation. We used

the same training regime as described in our previous work (Maldonado et al., 2017) to create pairwise training data from train-v2.0 using pairs of articles with differing relevance values. To train the model we used the Adam Optimizer (Kingma and Ba, 2014) with default parameters.

2.2 Details of the STNet

The Transformer Encoder. In order to facilitate the semantic soft-matching between words from a section heading path and words from a paragraph, the Transformer Encoder, depicted in Figure 3.b produces a sequence of vector encodings, o_1, \ldots, o_N representing the sequence of words in a section heading path, $w_1^s, \ldots w_N^s$. Each word w_i^s in a section heading path has an associated *n*dimensional word embedding (n = 300 in this work), $e(w_i^s) \in \mathbb{R}^n$, learned by GloVe (Pennington et al., 2014) on the full text of Wikipedia that captures fine-grained semantic and syntactic information about the word w_i^s .

However, it is not sufficient to use the GloVe embedding as a representation for the words in a section heading path since the same word may have a completely different meaning depending on its context (e.g. the word "turtle" in the Wikipedia articles "Sea Turtle" and "Turtle Ship"). Moreover, if a word is from the article title of a section heading path, it plays a different role when determining relevance than if it were from a section heading. To capture this sort of context information in our section path encoding, we use three separate Transformer Encoders (Vaswani et al., 2017) to produce an encoding of each section heading word that takes context information into account using self-attention. We train three separate Transformer Encoders to learn separate encodings of words from (1) a section heading; (2)an intermediate heading; and (3) an article title.

Each Transformer Encoder learns a contextualized encoding, o_i^s , for each word, w_i^s using a stack of 6 Transformer Encoder Blocks, TE_k for $k \in [1, \ldots, 6]$. The output of the k^{th} block for the i^{th} word, denoted as ebl_i^k , is given by:

$$ebl_{i}^{k} = ebl_{i}^{k-1} + TE_{k}(ebl^{k-1})_{i}$$
 (3)

$$TE_K(ebl^{k-1}) = LN(FFN(\hat{\mathbf{O}}) + \hat{\mathbf{O}})$$
(4)

$$\hat{\mathbf{O}} = LN(MHS(ebl^{k-1}) + ebl^{k-1})$$
(5)

where $LN(\cdot)$ is the Layer Normalization function (Ba et al., 2016), $FFN(\cdot)$ is a feed-forward network and $MHS(\cdot)$ is MultiHead self-attention. The feed-forward network, FFN, is a simple twolayer network comprised of ReLU and linear units:

$$FFN(x) = W_1^f ReLU(W_2^f x + b_2^f) + b_1^f$$
 (6)

where $W_1^f, W_2^f \in \mathbb{R}^{n \times n}$ are weight matrices and $b_1^f, b_2^f, \in \mathbb{R}^n$ are bias vectors. The MultiHead selfattention function, $MHS(\cdot)$, allows the model to jointly attend to information from different representation subspaces at different positions of a section heading path. This is accomplished by computing, in parallel, h different self-attention functions – referred to as attention *heads* – and combining the results of each head as follows:

$$MHS(\mathbf{X}) = [head_1, \dots, head_h]W^M \quad (7)$$

$$head_i = Attn(\mathbf{X}W_i^Q, \mathbf{X}W_i^K, \mathbf{X}W_i^V) \quad (8)$$

$$Attn(Q, K, V) = \operatorname{softmax}\left(\frac{QK^{\mathsf{T}}}{\sqrt{n}}\right)V \quad (9)$$

where $W^M \in \mathbb{R}^{n \times n}, W_i^Q, W_i^K, W_i^V \in \mathbb{R}^{n \times \frac{n}{h}}$ for $i \in [1, \ldots, h]$ are weight matrices, $[\cdot]$ is the concatenation operation, and n is hidden size of the model. In this work, we use h = 10 attention heads.

The final encoding of word w_i^s is given by $o_i^s = ebl_i^6$ and the input to the encoder blocks is given by the word embeddings combined with a positional mask, $ebl_i^0 = e(w_i^s) + PE_i$, where:

$$PE_{ij} = \begin{cases} \sin\left(\frac{i}{10000^{j/n}}\right) & \text{if j is even} \\ \cos\left(\frac{i}{10000^{j/n}}\right) & \text{if j is odd} \end{cases}$$
(10)

The final section path encoding is produced by concatenating the encodings of (1) the section heading, o^{s_1} , (2) the intermediate headings, o^{s_2} , and (3) the article title, o^{s_3} : $o^s = [o^{s_1}, o^{s_2}, o^{s_3}]$.

The Transformer Decoder. The role of the Transformer Decoder, illustrated in Figure 3.c is to perform a semantic soft-matching between each word from the paragraph text, p, and each word from the section heading path, s, using attention and to represent this soft matching into a single vector, a(s, p). The Transformer Decoder makes use of attention to determine which pairs of words help indicate relevance between a section heading path and a paragraph, even if the words are not the same (i.e. a hard match). Moreover, self-attention over the paragraph words can be used to contextualize the representations of each word, as in the Transformer Encoder.

Like the Transformer Encoder, the Transformer Decoder is comprised of a stack of 6 identical blocks, TD_k for $k \in [1, ldots, 6]$, where the output of the k^th block for the i_th paragraph word, denoted as dbl_i^k , is given by:

$$dbl_i^k = dbl_i^{k-1} + TD_k(dbl^k)_i \quad (11)$$

$$TD_k(dbl^{k-1}) = LN(FFN(\hat{\mathbf{O}}^2) + \hat{\mathbf{O}}^2) \quad (12)$$

$$\hat{\mathbf{O}}^2 = LN(MHA(\hat{\mathbf{O}}^1, \mathbf{O}^s) + \hat{\mathbf{O}}^1) \quad (13)$$

$$\hat{\mathbf{O}}^1 = LN(MHS(dbl^{k-1}) + dbl^{k-1}) \quad (14)$$

where $LN(\cdot)$, $FFN(\cdot)$, and $MHS(\cdot)$ correspond to Layer Normalization, feed-forward network, and MultiHead Self-attention as defined in the Transformer Encoder section, $MHA(\cdot, \cdot)$ is MultiHead Attention, and $\mathbf{O}^s = o_{\cdot}^s$ is a matrix comprised of the full section heading path encoding. MultiHead Attention computes an *h* attention weights over each pair of words from the section heading path and paragraph, one for each attention head. Like MultiHead Self-attention, MHA uses multiple attention heads to capture different aspects of the semantic matching in parallel. MHAis computed similarly to MHS, however it incorporates the section heading path encoding, \mathbf{O}^s :

$$MHA(\mathbf{X}, \mathbf{O}^{s}) = [head_{1}, \dots, head_{h}]W^{M}$$
(15)
$$head_{i} = Attn(\mathbf{X}W_{i}^{Q}, \mathbf{X}W_{i}^{K}, \mathbf{O}^{s}W_{i}^{V})$$
(16)

where where $W^M \in \mathbb{R}^{n \times n}, W^Q_i, W^K_i, W^V_i \in \mathbb{R}^{n \times \frac{n}{h}}$ for $i \in [1, \ldots, h]$ are weight matrices and $Attn(\cdot)$ is defined in Equation (9). Again, as in the Transformer Encoder, the input to the first Transformer Decoder block is given by the word embeddings of the paragraph words combined with a positional mask: $dbl^0_i = e(w^p_i) + PE_i$ where PE is defined in Equation 10.

In order to create the semantic matching vector, a(s, p), we combine the vectors produced by the final block of the Transformer Decoder using max pooling over the paragraph dimension and pass the pooled vector through a ReLU layer:

$$\mathcal{P} = \mathrm{maxpool}_0(dbl_i^6) \tag{17}$$

$$a(s,p) = ReLU(W^a \mathcal{P} + b^a) \tag{18}$$

where $W^a \in \mathbb{R}^{n \times n}$ is a weight matrix, $b^a \in \mathbb{R}^n$ is a bias vector, and $\max \text{pool}_0(\cdot)$ denotes max pooling over the 0 dimension. The intuition behind max pooling over the paragraph dimension is to select the strongest relevance signals among every paragraph word.

The Neural Similarity Module. The Neural Similarity module is based on the deep neural relevance model of the top-performing system from the 2017 TREC-CAR track which uses PACRR (Hui et al., 2017). PACRR operates in 4 steps: (1) it constructs a similarity matrix given a section path and a paragraph; (2) passes the similarity matrix through several convolutional layers and max-pooling layers, constructing a vector for each word in the section path representing the strongest similarity signals that word had with the entire paragraph; (3) concatenates each section path word vector with a *contextual vector*; and (4) combines each section path word vector into a single vector using a LSTM.

Given a section heading path, s, and a paragraph, p, the Neural Similarity Module will calculate a vector $\psi(s, p)$ representing the similarity between s and p. First, we construct a similarity matrix, $sim \in [-1,1]^{|s| \times |p|}$, where sim_{ij} is the cosine similarity between the word embeddings of the i^{th} word of the section heading path and the j^{th} word of the paragraph. We use q-1 different convolutional layers with square kernel sizes $2 \times 2, 3 \times 3, \dots, g \times g$ corresponding to bi-gram, tri-gram, ..., g-gram matching. All convolutional layers use f_c filters and have a stride size of (1, 1), resulting in g-1 tensors, $C^g \in \mathbb{R}^{|s| \times |p| \times f_c}$. Two pooling layers are used to highlight the strongest signals for each query term: max pooling over the filter dimension, and k-max pooling over the paragraph-word dimension, resulting in a single tensor $C^p \in \mathbb{R}^{|s| \times g - 1 \times k}$. This tensor is flattened into a series of (k + q - 1)-dimensional vectors, one for each word in the section heading path, s. Each of these vectors is concatenated with a contextual vector consisting of a three dimensional 1hot encoding of what kind of heading a section path word belonged to (section heading, intermediate heading, or article title) along with the normalized IDF of the word. Finally, the vectors are passed through a LSTM with hidden size n. The similarity encoding between section heading path s and paragraph p is given by the final hidden state of the LSTM, $\psi(s, p) \in \mathcal{R}^n$.

The Feature Encoder. To incorporate features extracted between a section s and a paragraph p, STNet encodes a set of features into a single vector, f(s, p), called the *feature encoding* using the same module as SANet from the CAPAR sys-

	Manual			Lenient		
Paragraph Ranker	MAP	R-Prec	NDCG	MAP	R-Prec	NDCG
STNet	0.2545	0.2789	0.4280	0.3125	0.3404	0.5293
STNet(-TRANS)	0.2152	0.2644	0.3774	0.2625	0.3321	0.4565
STNet(-SIM)	0.2150	0.2604	0.3780	0.2697	0.3398	0.4626

Table 1: Results for TRANS-CAPAR using Mean Average Precision (MAP), R-Precision (R-Prec), and Normalized Discounted Cumulative Gain (NDCG). Highest scores in bold.

tem (Maldonado et al., 2017). The Feature Encoder module, fully defined in Maldonado et al. (2017), composes the feature encoding, f(s, p) by using a series of 1-dimensional convolutional layers to combine a set of feature vectors for each word and entity in section heading path and the paragraph.

The Scorer. The Scorer learns a function that combines the feature encoding, f(s, p), similarity encoding, $\psi(s, p)$, and the semantic matching vector, a(s, p) to produce a real valued score representing how relevant a paragraph, p is to the section, s. Formally, the score $\phi(s, p)$ for a section s and a paragraph p is given by the following equation:

$$\phi(s,p) = \operatorname{elu}\left(v_s^T\left[a(s,p), f(s,p), \psi(s,p)\right]\right)$$
(19

where elu(·) is the Exponential Linear Unit activation function (Clevert et al., 2015) and $v_s^T \in \mathbb{R}^{|a(s,p)|+|f(s,p)|+|\psi(s,p)|}$ is a learned weight vector.

3 Results

The results for TRANS-CAPAR in the 2018 TREC CAR evaluation are summarized in Table 1. Two evaluation protocols are used to evaluate the paragraph retrieval task: (1) Manual Evaluation, and (2) Lenient Evaluation. The top 100 results for participant systems were pooled, and judgments were created by National Institute for Standards and Technology (NIST) judges assigning each paragraph a score of 3 (MUST BE MENTIONED), 2 (SHOULD BE MENTIONED), 1 (COULD BE MEN-TIONED), 0 (ROUGHLY ON TOPIC), -1 (NON-RELEVANT), or -2 (TRASH). In the lenient evaluation a paragraph is considered relevant if it is determined to be at least ROUGHLY ON TOPIC. However, in the manual evaluation paragraphs are only relevant if their relevance is at least COULD BE MENTIONED. For both the manual and lenient evaluations a paragraph is considered irrelevant if it has not been manually assessed for relevance.

In order to test our hypothesis that the Transformer and Neural Similarity module capture semantic and similarity information from the question and paragraph, we evaluated the STNet relevance model against two alternate configurations in which we omit either the Transformer Encoder and Decoder (-TRANS) or the Neural Similarity module (-SIM). The three STNet configurations are evaluated using three standard IR metrics: Mean Average Precision (MAP), R-Precision (R-Prec), and Normalized Discounted Cumulative Gain (NDCG). Clearly, the STNet relevance model performs best on all metrics for both manual and lenient evaluations. Interestingly, the two alternate configurations of STNet perform remarkably similar to one another, with absolute differences of 0.0002, 0.004, and 0.006 in MAP, R-Precision, and NDCG, respectively, on the manual evaluation.

4 Discussion

Complex answer retrieval is a challenging task and it is clear from these results that further research is necessary to address its complexities. However, with a 26.6% relative increase in MAP over the top performing system in TREC CAR 2017, TRANS-CAPAR shows promise.

The fact that STNet that uses both the Transformer Encoder/Decoder and the Neural Similarity module outperforms both alternate configurations that use one or the other suggests that the two approaches are indeed complimentary. We believe this is because, when used together, the Neural Similarity module is able to capture similarity signals while the Transformer captures semantic information and discourse context. For example, consider the test topic "Air pollution/The *Clean Air Act*". For this query, the top result of the STNet-TRANS configuration which uses the Neural Similarity module, but not the Transformer, is a paragraph about 'New Source Performance Standards' because that paragraph contains the n-gram "Clean Air Act" several times. In contrast, the top ranked result of the STNet-SIM configuration which uses the Transformer, but not the Neural Similarity module is a relevant paragraph that contains the n-gram "Clean Air Act" only once, but it is clear from the discourse context that it is the subject of the paragraph. Conversely, for the test topic "*Relative ages of rocks/Law of Lateral Continuity*", the top ranked result returned by STNet–SIM about the continuation of Roman law through the ages, but STNet–TRANS is able to correctly rank a relevant paragraph by identifying the similarity between the word '*rocks*' from the question and the words '*geologic*' and '*outcrop*' in the paragraph. For both of these questions, STNet using both the Transformer and Neural Similarity module is able to correctly rank the better paragraph in both cases.

5 Conclusion

The TRANS-CAPAR system used in the official evaluation of the 2018 TREC CAR track considered that answering complex questions by providing a ranked list of paragraphs performs best when taking into account both the similarity between the question paths and the paragraphs as well as the semantics that they share. However, capturing semantics between questions and paragraphs is notoriously difficult. We developed a neural architecture that encodes the semantics of the question and decodes it against the paragraph while also taking into account their similarity. The neural architecture used the encoder and decoder implemented by the Transformer (Vaswani et al., 2017) architecture to account for "deep" semantics. The results demonstrate that both similarity and deep semantics are useful in ranking.

References

- Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.
- Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. 2015. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*.
- Kai Hui, Andrew Yates, Klaus Berberich, and Gerard de Melo. 2017. Pacrr: A position-aware neural ir model for relevance matching. *arXiv preprint arXiv:1704.03940*.
- Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

- Sean MacAvaney, Andrew Yates, Arman Cohan, Luca Soldaini, Kai Hui, Nazli Goharian, and Ophir Frieder. 2018. Characterizing question facets for complex answer retrieval. *arXiv preprint arXiv:1805.00791*.
- Ramon Maldonado, Stuart Taylor, and Harabagiu Sanda M. 2017. Utd hltri at trec 2017: Complex answer retrieval track. In *Text REtrieval Conference 2017*.
- Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.