Optimizing F1-Score for Text Spans Extraction in Question Answering Systems

Mohamed Reda Bouadjenek[◇], Scott Sanner[♦], Rohan Man Amatya[◇], Asef Nazari[◇], Imran Razzak[◇]
[◇]School of Information Technology, Deakin University, Waurn Ponds Campus, Geelong, Australia
[♦]Department of Mechanical and Industrial Engineering, The University of Toronto, ON, Canada
[◇]{reda.bouadjenek,rmamatya,asef.nazari,imran.razzak}@deakin.edu.au, [♦]ssanner@mie.utoronto.ca

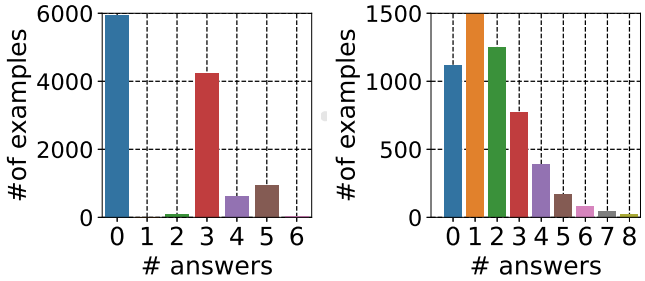
ABSTRACT

Current question answering systems are based on complex deep neural network architectures where often, the output consists of two classification layers that are used to predict simultaneously the start and end positions of the answer in the input passage. However, we argue in this paper that this approach suffers from several drawbacks. First, it does not optimize the metrics that are used to evaluate the model's performance, e.g., *F1-Score* or *exact-match*. Second, this approach does not allow encoding all possible answers during training, and during prediction, it only focuses on one possible answer and does not allow the extraction of other candidate answers. Lastly, this approach does not prevent the case where the token with the highest probability for being the answer start token comes after the token with the highest probability for being the answer end token. This paper addresses these deficiencies by proposing a novel approach for text spans extraction, which consists of a single classification output layer followed by an independent optimization module. This 2-step approach aims first to optimize *exact-match* and then extracts a text span by maximizing an *expected* F1-Score (EF1) objective. Specifically, we contribute with a Mixed Integer Linear Programming (MILP) formulation to optimize EF1 subject to parameterized constraints for text spans extraction. The proposed approach allows the extraction of multiple possible answers to a question from a passage and to identify if a span of text is a valid answer and so the answerability of the question given the passage. We demonstrate the effectiveness of our approach with extensive experimental evaluations and a comparison against existing baselines on the SQuAD1.1, SQuAD2.0, and NewsQA datasets.

Keywords: Question Answering, Optimization, MILP, Text Spans Extraction.

1 INTRODUCTION

Question Answering (QA) is a basic Natural Language Processing task that consists of automatically inferring the answer to a question from a given passage of text. Supported by various large-scale QA datasets [1–5], the QA task has allowed remarkable advancements in Machine Reading Comprehension (MRC) with important applications in conversational search [6], conversational recommendation [7], and customer service support [8].



(a) SQuAD2.0. (b) NewsQA.

Figure 1: Distribution of the number of answers in the SQuAD2.0 and NewsQA datasets.

One important hypothesis behind current approaches in QA is that the answer to a question is a segment of text, or a *span*, from the corresponding reading passage [1, 3]. Hence, current solutions are based on a complex deep neural network architecture that takes a question-passage pair as input to predict the start and end positions of the answer in the passage. Specifically, as illustrated in Figure 2, the output consists of two independent classification layers that are trained simultaneously to obtain a probability distribution over the start and end tokens of the answer in the passage.

However, we argue in this paper that this solution suffers from a number of deficiencies. First, this approach does not optimize the metrics that are used to evaluate the model performance [1], e.g., *F1-Score* or *Exact Match*. Second, as illustrated in Figure 1, most of the examples in the official development set of SQuAD2.0 [3] and the NewsQA [2] datasets have more than one possible answer in a passage – some have even five or six possible answers. However, the conventional approach as illustrated in Figure 2 does not allow encoding all possible answers during training, and during prediction, it only focuses on one possible answer and does not allow the extraction of other candidate answers. Finally, we note that this approach to span extraction does not prevent the case where the token with the highest probability for being the answer start token comes after the token with the highest probability for being the answer end token. Empirically, we have noticed with our baselines that this happens in about 5% to 10% of the cases.

To deal with the aforementioned problems, we propose a novel model agnostic approach for text spans extraction. Briefly, our

Unpublished working draft. Not for distribution.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
Conference'17, July 2017, Washington, DC, USA
© 2021 Association for Computing Machinery.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

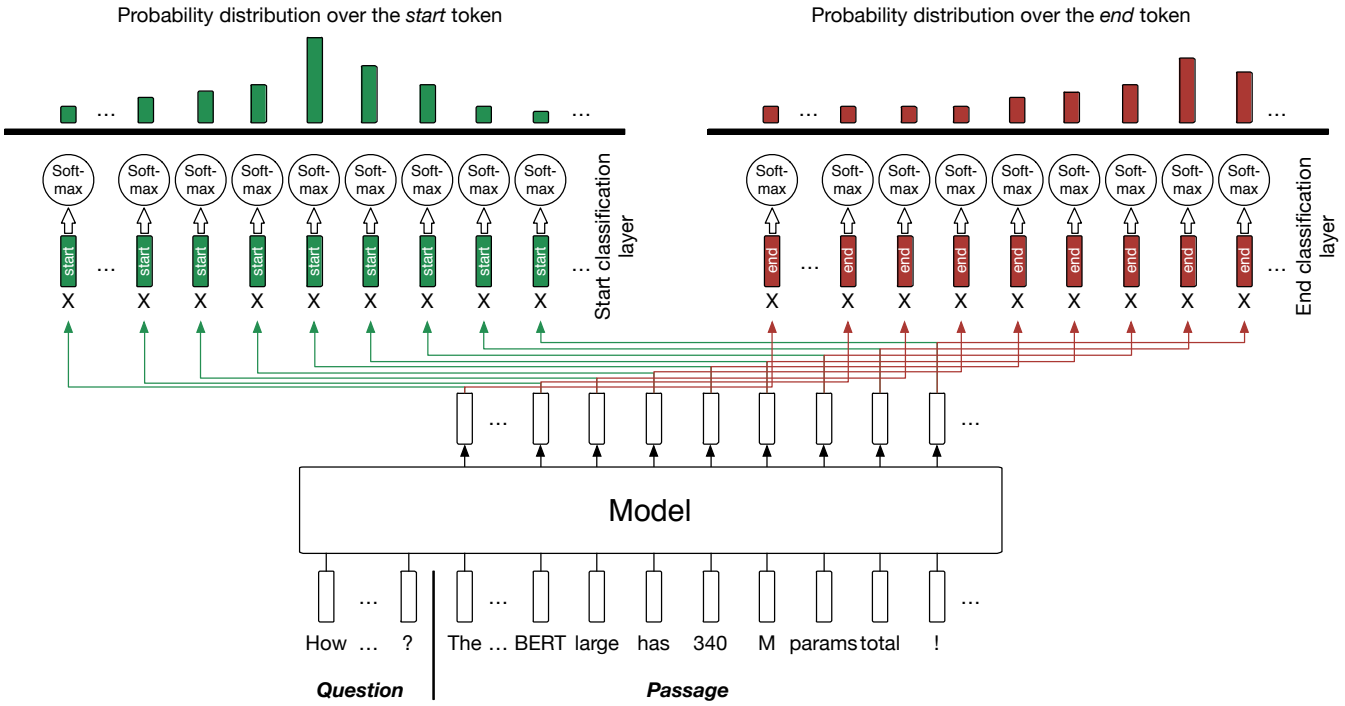


Figure 2: Architecture of a conventional Question Answering System [9].

double-edged sword method is a 2-step approach that consists of: (i) a simplified architecture with a single classification output layer that aims to optimize *exact-match*, (ii) followed by an *independent* optimization module that extracts a text span by maximizing an *expected* F1-Score (EF1) objective. Specifically, we contribute with a Mixed Integer Linear Programming (MILP) formulation to optimize EF1 subject to parameterized constraints for text spans extraction. Also, our proposed method allows the extraction of multiple possible answers to a question from a passage, to identify if a span of text is a valid answer, and so to decide the answerability of the question given the passage. We demonstrate the effectiveness of our approach with extensive experimental evaluations and a comparison against existing baselines on the SQuAD1.1, SQuAD2.0, and NewsQA datasets. The **key contributions** of this paper are as follows:

- We propose a novel model agnostic approach for text spans extraction that aims to optimize the two main metrics for evaluating QA systems, i.e., exact-match and F1-Score. The proposed approach allows encoding all possible answers during training, and during prediction, it permits to extract multiple possible answers, to identify if a span of text is a valid answer and if a question is answerable.
- We propose new expected metrics for optimization based on conventional IR metrics, i.e., precision, recall, and F1-Score.
- We contribute with a Mixed Integer Linear Programming (MILP) formulation to optimize EF1 subject to parameterized constraints for text spans extraction.
- We present a thorough evaluation on three different datasets to show the effectiveness of our methods against existing baselines. We experimentally demonstrate that the expected F1-Score metric is a good surrogate of F1-Score.

The rest of this paper is organized as follows: : in Section 2 we present the related work. Next, in Section 3 we present the text spans extraction approach we propose and the optimization framework. In Section 4 we describe our experimental setup before discussing our results in Section 5. Finally, we conclude and provide some future directions in Section 6.

2 RELATED WORK

There is a substantial body of research related to Question Answering systems. Below, we first survey QA systems before discussing in-depth spans extraction methods used in QA systems.

2.1 QA systems

Question answering systems are often meant to answer factoid questions (including definition, fact, list, Why, How, hypothetical, semantically constrained, and cross-lingual questions) [10]. Often, QA systems are classified according to their field of application, the field in which they operate and answer questions. In particular, there are mainly two classes of QA systems [10, 11]: Closed-domain QA Systems (cdQA) and Open-domain QA Systems (odQA). The earliest QA systems (BASEBALL [12] and LUNAR [13]) were developed in the 60s and 70s and were highly domain-specific (closed domain question answering system). A cdQA system deals with

Unpublished working draft. Not for distribution.

questions under a specific domain such as healthcare [14–16], legal [17, 18], education [19, 20], etc. Thus, it mainly refers to the situation where only limited types of questions are accepted and are not really viable when dealing with the public. However, language is dynamic, and people can use infinite number of ways to ask a question. Therefore, QA systems have then paved the way for a new research on factoid questions (questions that have enumeration of short answers and are answered with simple facts), which are often treated as search queries to extract the most relevant answer.

Unlike a cdQA system, an open-domain question answering (odQA) system works on text lacking the relevant context for any arbitrarily asked factual question and return the answer in the form of short texts rather than a list of relevant documents. Ideally, an odQA system should be able to sift through a large text corpus to find the answer. Historically, odQA systems required specialized complex pipelined systems consisting of several machine learned and hand-crafted modules [21] which has recently been shifted to end-to-end deep neural networks [22, 23].

Finally, approaches and techniques of QA systems are often based on [24]: (i) linguistic and natural language processing techniques such as tokenization, Part Of Speech tagging and parsing (e.g., BASEBALL [12], LUNAR [13], ELIZA [25], GUS [26], etc.); (ii) statistical approaches that leverage large amount of data by applying statistical methods such as support vector machine [27, 28], similarity measures [29], and maximum entropy models [30, 31]; and (iii) pattern matching techniques including surface text patterns [32, 33] and templates [34, 35] for generating answers.

2.2 Span extraction in QA systems

Current QA systems tend to have a similar architecture to the one depicted in Figure 2 [9, 36–43]. Specifically, they almost all rely on complex deep neural network architectures, where the output consists of two independent classification layers trained simultaneously. The parameters of these two classification layers are a start parameter vector $S \in \mathbb{R}^H$ and an end parameter vector $E \in \mathbb{R}^H$. The probability of a word j being the start of the answer span is computed as a dot product between T_j, a contextualized word representation, and S followed by a softmax over all of the words in the paragraph: $P(j) = \frac{e^{S \cdot T_j}}{\sum_i e^{S \cdot T_i}}$. The analogous formula is used for the end of the answer span. The score of a candidate span from position i to position j is often defined as $P(i) + P(j)$, and the maximum scoring span where $j \geq i$ is used as a prediction [9].

Finally, the authors in [44] have proposed Pointer Net, a sequence-to-sequence model (encoder-decoder architecture) that constrains the output tokens to be from the input sequences. Pointer Net has been used by [45] who proposed an end-to-end neural architecture for QA. Comparing to the method that we propose in this paper, we argue that the Pointer Net [45] still suffers from the problem that it cannot trivially extract all possible answers in a passage and that the decoder may generate a span that is not part of the input passage. The two methods proposed in [45] are used as baselines for comparison in Section 5.

3 TEXT SPANS EXTRACTION APPROACH

In this section, we first describe the architecture of our approach, then we provide details of the optimization framework we propose.

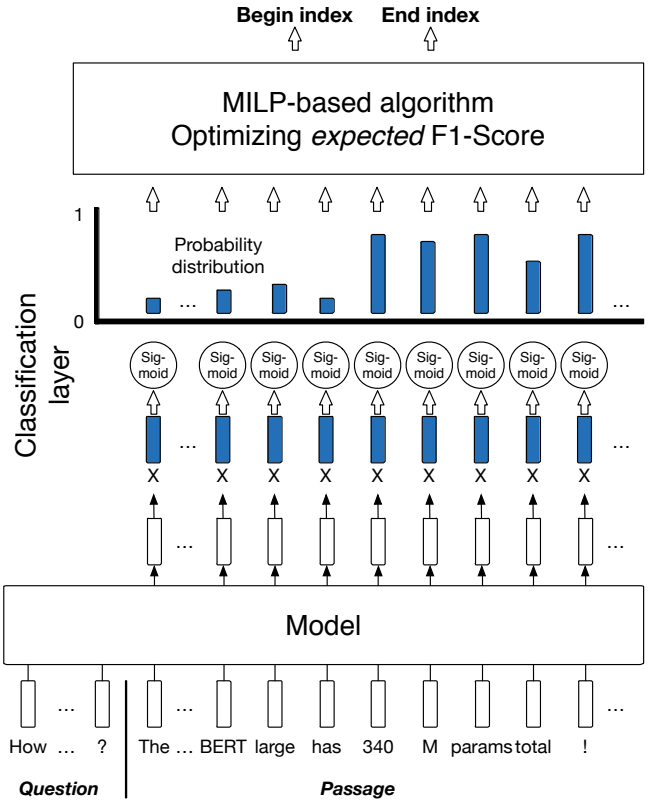


Figure 3: Architecture of the proposed approach.

3.1 Architecture overview

Briefly, our approach relies on a two-step process of deriving a contextualized word representation for classification, and a MILP-based optimization approach for answer extraction. The architecture of the approach we propose is quite straightforward. As shown in Figure 3, the first step consists of the use a deep neural network architecture, for instance RNN, Bi-RNN, Attention Model, ELMo [46], or BERT [9], that can take a question-passage pair as input, and that can output T_j, a contextualized word representation for each token j in the passage. Next, we only introduce a binary classification layer with a parameter vector $C \in \mathbb{R}^H$ during fine-tuning (H is the size of the hidden vector). The probability $P(j)$ of word j being part of the answer span is computed as a dot product between T_j and C followed by a sigmoid: $P(j) = \frac{1}{1+e^{-C \cdot T_j}}$. The training objective is to maximize the log-likelihood of the correct answer in a passage. We note that this architecture allows encoding all possible answers to a question during training time as it formulates the task as a multi-label classification problem.

The second step is an optimization framework used to extract the answer span. We note that the method we propose here is model agnostic since the two processes are independent from each other. A detailed description of the optimization framework is provided in the next subsection.

Unpublished working draft. Not for distribution.

3.2 Optimization framework

We argue that the (expected) F1-Score of a substring is the only standard criteria that balances all of substring desiderata and is hence the objective we should optimize. To proceed with the formal derivation, we adopt the Boolean relevance framework in information retrieval and thus assume that a token j has a ground truth relevance assessment $B(j)$ available at evaluation time.

Because selecting a substring implies a Boolean selection model (a substring either contains or does not contain tokens) and we have a probabilistic estimate of relevance $P(j)$ for each token, we propose to evaluate *expected* variants of standard precision, recall, and F1-score of these substrings.

However, we note that precision and recall alone can be trivially optimized by undesired solutions. That is, the substring that selects all tokens would trivially maximize (expected) recall. Similarly, the substring that selects the highest probability singleton token would maximize expected precision. This leaves expected F1-score as the most meaningful and logical measure among those commonly used in Boolean information retrieval that does not have an undesired solution.

3.3 Deriving *expected* F1-Score (EF1)

To formally define *expected* F1-Score, we first begin with definitions of expected precision and expected recall. Given a substring s of a string S ($s = S[b, e]$, $1 <= b <= e <= m$, and $|S| = m$), the precision of s is defined as follows:

$$P(s) = \frac{\sum_{j \in s} B(j)}{|s|} = \frac{\sum_{j=1}^{m} B(j)I(j)}{\sum_{j=1}^{m} I(j)}$$

where the two variables $I(j) \in \{0, 1\}$ and $B(j) \in \{0, 1\}$ are associated with each token j of S:

- $I(j)$ is an indicator referring to whether a token j is part of s (true = 1);
- $B(j)$ is a Boolean random variable indicating the (ground truth) relevance of a token j (relevant = 1).

Given that $B(j)$ is a Boolean random variable, we can take the expectation of $P(s)$ leading to the following definition of *expected precision* $EP(s)$:

$$\begin{aligned} EP(s) &= \mathbb{E}_{\mathbb{P}}\left[\frac{\sum_{j=1}^{m} B(j)I(j)}{\sum_{j=1}^{m} I(j)}\right] \\ &= \frac{\sum_{j=1}^{m} \mathbb{E}_{\mathbb{S}}[B(j)]I(j)}{\sum_{j=1}^{m} I(j)} = \frac{\sum_{j=1}^{m} P(j)I(j)}{\sum_{j=1}^{m} I(j)} \end{aligned} \tag{1}$$

where the variable $P(j) \in [0, 1]$ is the *probability* relevance of a token j.

Similarly the recall of a substring s is defined as:

$$R(s) = \frac{\sum_{j \in s} B(j)}{\sum_{j \in S} B(j)} = \frac{\sum_{j=1}^{m} B(j)I(j)}{\sum_{j=1}^{m} B(j)}$$

Taking a first order Taylor expansion, we have the following expectation approximation $\mathbb{E}(X/Y) \approx \mathbb{E}(X)/\mathbb{E}(Y)$ for two dependent random variables X and Y [47]. Hence, we can now define an *approximated expected recall* as follows:

$$\begin{aligned} ER(s) &= \mathbb{E}_{\mathbb{P}}\left[\frac{\sum_{j=1}^{m} B(j)I(j)}{\sum_{j=1}^{m} B(j)}\right] \\ &\approx \frac{\sum_{j=1}^{m} \mathbb{E}_{\mathbb{P}}[B(j)]I(j)}{\sum_{j=1}^{m} \mathbb{E}_{\mathbb{P}}[B(j)]} = \frac{\sum_{j=1}^{m} P(j)I(j)}{\sum_{j=1}^{m} P(j)} \end{aligned}$$

Finally, we define the *approximated expected F1-Score* (EF1) using the *expected precision* and the *approximated expected recall* as follows:

$$\begin{aligned} EF1(E) &\approx \frac{2 \times EP \times ER}{EP + ER} \\ &= \frac{2 \times \sum_{j=1}^{m} P(j)I(j)}{\sum_{j=1}^{m} I(j) + \sum_{j=1}^{m} P(j)}. \end{aligned}$$

3.4 Fractional MILP formulation

We begin by reformulating the ***Expected* F1-score (EF1)** objective to prepare for further optimization steps by replacing the global sum of scores of all tokens with a constant $C = \sum_{j=1}^{m} P(j)$:

$$\begin{aligned} EF1 &= \frac{2 \times \sum_{j=1}^{m} P(j)I(j)}{\sum_{j=1}^{m} I(j) + \sum_{j=1}^{m} P(j)} \\ &= \frac{2 \times \sum_{j=1}^{m} P(j)I(j)}{\sum_{j=1}^{m} I(j) + C}. \end{aligned}$$

In order to obtain the substring s with the optimal ***Expected* F1-score**, we define the following fractional MILP:

$$\begin{aligned} &\underset{I(j)}{\text{maximize}} && \frac{\sum_{j=1}^{m} P(j)I(j)}{\sum_{j=1}^{m} I(j) + C} \\ &s.t. && \textit{\# Interval selection contraints.} \\ & && 1 <= b <= e <= m \\ & && I(j) = \begin{cases} 1, & \text{if } (b \leqslant j \leqslant e) \\ 0, & \text{otherwise.} \end{cases} \end{aligned}$$

3.5 Transformation to a MILP

While there are no direct solvers for fractional MILPs, we can transform the former equation into a pure MILP form for which we have efficient solvers. To do this, we use the Charnes-Cooper method [48] and Glover linearization method [49] with big-M constraints, where auxiliary variables $w(j)$ and u are introduced[1]. Here, $w(j)$ is defined as $w(j) = I(j) \times u$ with u defined as follows:

$$u = \frac{1}{\sum_{j=1}^{m} I(j) + C} \tag{2}$$

[1] https://optimization.mccormick.northwestern.edu/index.php/Mixed-integer_linear_fractional_programming_(MILFP)

Unpublished working draft. Not for distribution.

Thus, the EF1 optimization problem is able to be transformed into the following MILP problem:

$$\underset{w,u,I,b,e}{\text{maximize}} \quad \sum_{j=1}^{m} P(j)w(j)$$

$$s.t. \quad \sum_{j=1}^{m} w(j) + uC = 1$$

$$w(j) \leqslant u, \quad w(j) \leqslant M \times I(j)$$

$$w(j) \geqslant u - M \times [1 - I(j)]$$

$$u > 0, \quad I(j) \in \{0, 1\}, \quad w(j) \geqslant 0$$

 # Interval selection constraints.

$$1 <= b <= e <= m$$

$$I(j) = \begin{cases} 1, & \text{if } (b \leqslant j \leqslant e) \\ 0, & \text{otherwise.} \end{cases}$$

3.6 Threshold-based Answerable Verification

Following previous research [9, 41, 50–52], we also adopt a threshold based answerable verification (TAV) approach, which consists of a simple heuristic strategy to decide whether a question is answerable according to the predicted span of text. Specifically, we propose to use a threshold θ to decide if a question is answerable as follows: given the *first* candidate answer s, if $EP(s) > \theta$ then $s \in P$ is an answer to Q. Otherwise P does not contain an answer to Q – we recall *EP(s)* is the *expected precision* in Equation 1. The threshold θ is selected on the validation set to maximize F1-Score.

3.7 Multiple answer selection wrapper

In practice, a single substring s chosen by the previously described MILP-based algorithm will provide the user with one possible answer to a question. However, as shown in Figure 1, a passage can contain multiple answers to a given question. Here, we provide a greedy approach for providing a ranked list of such answers; we leave it to future work to develop improved answer extraction and ranking methods for multiple answers.

The algorithm itself is quite simple and simply wraps the previous algorithm. After the first substring is selected, all token elements in that substring have their scores $P(j)$ zeroed out – we recall $P(j)$ is the probability of relevance of token j. The algorithm is then run again, where it will inherently focus on a different substring. The algorithm stops when the new candidate doesn't pass the TAV test.

4 EXPERIMENTAL SETUP

Datasets: We evaluate the approach we propose on the Stanford Question Answering Datasets SQuAD1.1 [1] and SQuAD2.0 [3], and the NewsQA dateset [2].

In SQuAD1.1, the passages come from approximately 500 Wikipedia articles and the questions and answers were obtained by crowdsourcing. The crowdsourced workers were asked to read a passage (a paragraph), come up with questions, then mark the answer span. SQuAD2.0 combines the questions in SQuAD1.1 with over 50,000 unanswerable questions written adversarially by crowdworkers to look similar to answerable ones. The official development set is used to tune the hyperparameters of the neural network, and the original training set is splitted into training set and test by articles to avoid data leakage. The NewsQA dataset contains over 100,000 human-generated question-answer pairs collected by crowdworkers who supplied questions and answers based on a set of over 10,000 news articles from CNN, with answers consisting of spans of text from the corresponding articles. Details of the train/val/test sets used are given in Table 1.

Table 1: Description of datasets.

Dataset	SQuAD1.1	SQuAD2.0	NewsQA
Training set			
#articles	419	406	11,469
#paragraphs	17,925	17,407	11,469
#questions	83,048	118,264	97,309
#unanswerable questions	0	39,168	20,753
Validation set			
#articles	48	36	638
#paragraphs	2,067	1,628	638
#questions	10,570	12,055	5,456
#unanswerable questions	0	4,330	1,115
Test set			
#articles	23	35	637
#paragraphs	971	1,204	637
#questions	4,551	11,873	5,412
#unanswerable questions	0	5,945	1,120

Metrics: The performance is measured using the following evaluation metrics: (i) Exact Match (EM), which measures the percentage of span predictions that matched the ground truth answer exactly, (ii) F1-score, (iii) Precision, and (iv) Recall. Also, as suggested in [3], abstaining answering negative examples gives a score of 1, and any other response gives 0, for all metrics.

Implementation details: The model we have used in our implementation is BERT_{BASE} [9], which has 12 Transformer blocks, a hidden size $H = 768$, and 12 self-attention heads. The max sequence length (question-passage pair) was set to 384 tokens. The final classification layer was followed by a dropout layer. The classification dropout, the hidden dropout, and the attention dropout values were selected from a search within the discrete set {0.0, 0.2,0.4,0.6}, and the learning rate was selected from a search within the discrete set $\{9 \cdot 10^{-06}, 10^{-05}, 3 \cdot 10^{-05}, 5 \cdot 10^{-05}, 7 \cdot 10^{-05}, 9 \cdot 10^{-05}, 10^{-04}\}$. These parameters were selected by optimizing the F1-Score over the validation set using early stopping over the validation loss, before reporting the final results with the best parameters on the test set. We fine-tune the hyperparameters using early stopping and a batch size of 128 on the Google Colab platform with TPU.

5 EXPERIMENTAL EVALUATION

In this section, we report and discuss the main results we obtained in an offline evaluation and then a comparison with the baselines.

5.1 Offline evaluation

We first wish to understand how our MILP-based algorithm performs as we vary properties of the data and relevance score noise. Because we aim to evaluate the impact of noisy relevance evaluations on performance, it is critical to explicitly control noise levels,

Unpublished working draft. Not for distribution.

which we achieve through a noisy corruption of ground truth. In particular, we study and vary the following three properties:

Sequence length: We randomly select passages in our test set, and then, we explicitly vary their size by trimming/padding. We evaluate our approach with a sequence length of {10, 20, 30, 50, 100, 150, 200, 300, 400, 500}.

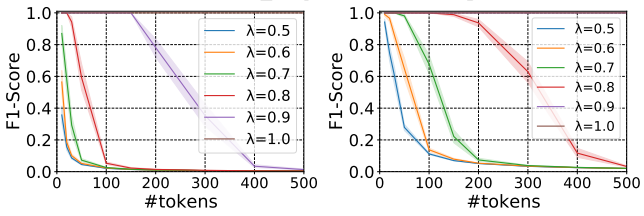
Level of noise: Once a passage is set to the right length, we assign for each token j the probability $P(j)$ to indicate its relevance probability by introducing a random noise signal as follows: $P(j) = \lambda \times B(j) + (1-\lambda) \times \text{rand}()$, where $B(j)$ is the boolean ground truth relevance, rand() is a random noise value chosen with uniform distribution in the range $[0, 1]$, and λ is a weighting parameter ($0.5 \leq \lambda \leq 1$) that controls the signal-to-noise ratio in the final probability value. Note that for $\lambda = 1$, $P(j)$ is a perfect predictor of the ground truth probability, whereas for $\lambda = 0.5$, $P(j)$ is extremely noisy (i.e., the signal-to-noise ratio is 1).

Size of answer: We consider passages that contain different answer lengths with values in the discrete set {1, 2, 3, 4, 5}.

Each evaluation was carried out by randomly selecting over 50 passages from our test set according to the designated sequence length and answer size. We report the average ground truth F1-Score (i.e., ground truth is known in the experimental setting), with 95% confidence intervals. We report and discuss the main results of the experimental evaluation, considering both the accuracy and the effectiveness of our MILP-based algorithm. The results shown are obtained on a MacBook Pro with a 2.8GHz Intel Core i7 CPU and 16GB 2133 MHz LPDDR3 of RAM, running MacOS X Mojave v10.14.6.

5.1.1 Performance analysis. First, we report experimental results under the previous settings in terms of F1-Score to analyze the performance of our optimization algorithm.

Varying sequence length: Here we aim to understand how our optimization algorithm performs as the amount of data varies for differing noise levels. This analysis is shown in Figure 4 while fixing the answer length to 1 and 5, using multiple values of λ, and varying the length of the sequence.

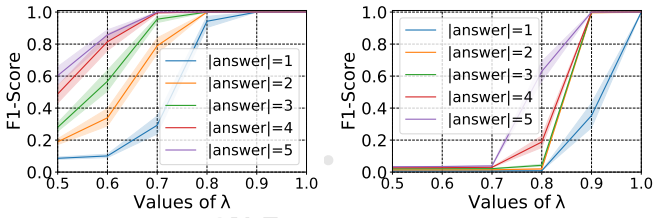


(a) Answer size = 1. **(b) Answer size = 5.**

Figure 4: Performance for different sequence lengths.

At first glance, we observe that for almost all values of noise and regardless of the answer length, the longer is the sequence, the lower is the performance. Second, we note that for $\lambda = 1$, our approach finds the optimal solution, and hence the F1-Score value is maximized to 1. Finally, we observe that for long answers, it is easier to optimize EF1 and therefore to obtain high values of F1-Score.

Varying relevance noise (λ): Here we aim to understand how our optimization algorithm performs as the amount of noise λ (defined previously) in the relevance prediction varies. The results of this analysis are shown in Figure 5 while fixing the sequence length to 30 and 300, using multiple answer lengths, and varying λ.

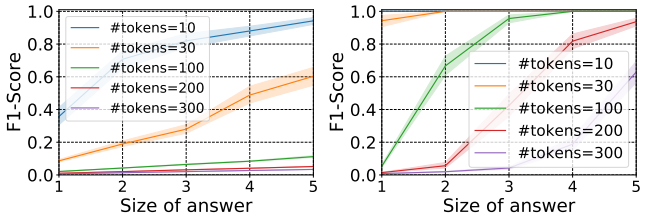


(a) #tokens = 30. **(b) #tokens = 300.**

Figure 5: Performance for different values of λ.

Briefly, we observe that the problem becomes easier (higher F1-Score value) as λ increases because EF1 becomes closer to the ground truth F1-Score. We also observe that the problem is easier for short sequences even with high noise (Figure 5a), i.e., low values of λ. Finally, we conclude that in general, our optimization algorithm is able to achieve decent F1-Score performance even in the presence of noise.

Varying size of answers: Here we aim to understand how our optimization algorithm performs as we vary the answer lengths in passages. Figures 6 shows the results of this analysis while fixing the value of λ to 0.5 and 1.0, using different sequence lengths, and varying the answer length.

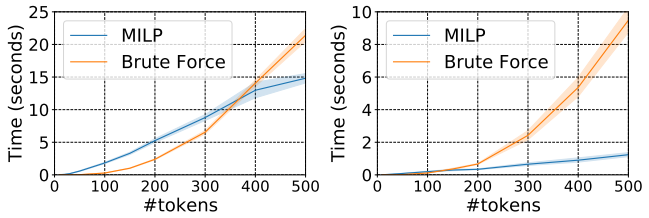


(a) $\lambda = 0.5$. **(b)** $\lambda = 0.8$.

Figure 6: Performance for different answer lengths.

We observe that the problem becomes easier as the answer increases as well as the value of λ increases (i.e., noise decreases). This is explained by the fact that a long answer allows the optimization algorithm to select a substring with a large number of positive tokens thus maximizing F1-Score. On the other hand, a low value of λ tends to lead the algorithm to select all tokens in the sequence due to the level of noise, while a high value of λ tends to lead the algorithm to select the actual positive tokens and thus maximizing F1-Score.

Unpublished working draft. Not for distribution.



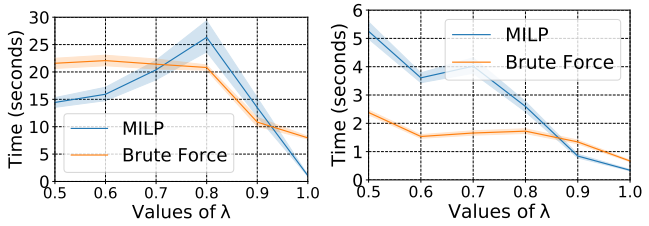
(a) $\lambda = 0.5$. **(b)** $\lambda = 1.0$.

Figure 7: Time complexity for different sequence lengths (Answer size = 5).

5.1.2 Time complexity analysis. We now report experimental results under the previous settings in terms of time complexity to analyze the scalability of our optimization approach. Here, we compare our MILP-based solution to the Brute Force search approach for reference.

Varying sequence length: A critical question for scalability is how time complexity of substring optimization scales vs. the sequence length. Figure 7 reports the results of this analysis while fixing the value of λ to 0.5 and 1.0, and the answer size to five. Naturally, the problem becomes harder as the sequence length increases. Overall, we critically notice that our MILP-based solution tends to have a lower time complexity than the Brute Force search approach for low noise (high λ). In particular, we observe that our MILP-based solution takes about one second to find the optimal solution for a sequence of 500 tokens with no noise. In contrast, finding the optimal solution for the same sequence with high noise takes about 15 seconds.

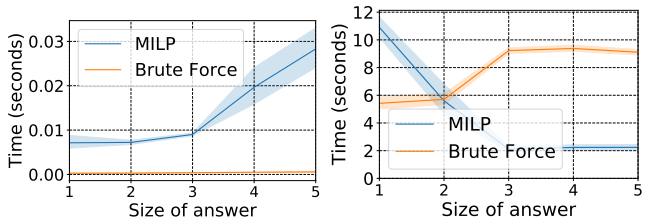
Varying relevance noise (λ): Another question is how noise levels in relevance prediction affect the time complexity. The results of this analysis are shown in Figure 8 while fixing the answer size to 2 and 5, and the sequence length to 500 tokens.



(a) Answer size = 2. **(b) Answer size = 5.**

Figure 8: Time complexity for different values of λ (#tokens=500).

In brief, we first notice that for this low values of λ the MILP-based solution has a high time complexity (up to 15 seconds to find the optimal solution for $\lambda = 0.5$ and an answer size of two tokens). Moreover, we observe that this time complexity reduces as the value of λ increases. We explain that by the fact that the MILP-based algorithm needs to make a large number of iterations to find the best solution with high noise because the size of feasible region is increased.



(a) #token=10. **(b) #token=400.**

Figure 9: Time complexity for different answer sizes ($\lambda = 0.9$).

Varying size of answer: Finally we ask how changes in the answer size affect the time complexity. Figure 9 show the results of this analysis while fixing the the sequence length to 10 and 400 tokens, and the value of λ to 0.9.

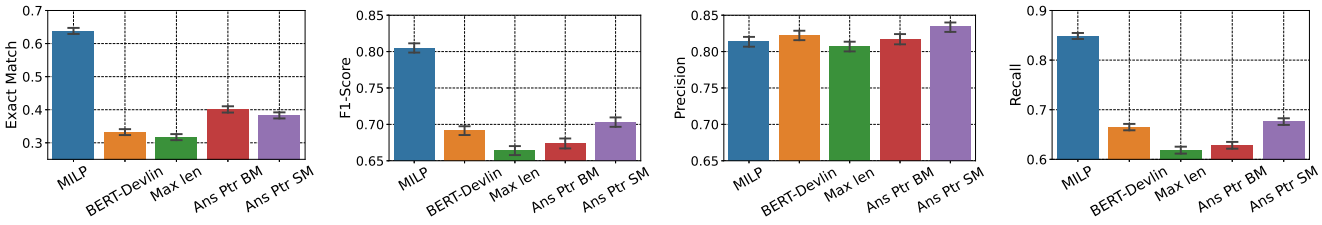
At first glance, we observe that for short sequences, a long answer tends to increase the complexity of our MILP-based solution. We explain this by the fact that the optimization algorithm tends to explore all possibilities to find the optimal solution as the size of feasible region is increased. In contrast, for long sequences, a long answer tends to decrease the complexity of our MILP-based solution. We explain this by the fact that in this situation, the solver can easily identify the optimal substring by eliminating irrelevant parts of the sequence.

5.2 Comparison against the baselines

In order to show the effectiveness of our approach, we compare its performance with the following two methods:

- **BERT-Devlin**: this method is the architecture depicted in Figure 2 and is described in [9]. This method uses the TAV in [9] to identify (un)answerable questions. We used the implementation released in: https://keras.io/examples/nlp/text_extraction_with_bert/. We have tuned the hyperparameters on our validation set and we've obtained similar hyperparameters to those described in [9].
- **Max len**: this method uses the neural network architecture depicted in Figure 3 but does not use our optimization framework. Instead, we convert the predicted probability $P(j)$ of each token j to 0 if $P(j) \leq 0.5$ or 1 otherwise. Then, we select the longest sequence of 1's as the answer to the question. This baseline allows to demonstrate the benefit of the MILP-based optimization framework we have proposed in Section 3.2. This methods uses a TAV by averaging over $P(j)$ in a candidate answer to identify (un)answerable questions.
- **Ans Ptr SM**: this method is the sequence model output depicted in Figure 1.a in [45]. Specifically, this method uses Pointer Net to treat an answer as a sequence of tokens from the input passage but ignores the fact that these tokens are consecutive in the original passage. This methods uses a TAV by averaging over $P(j)$ in a candidate answer to identify (un)answerable questions.

Unpublished working draft. Not for distribution.

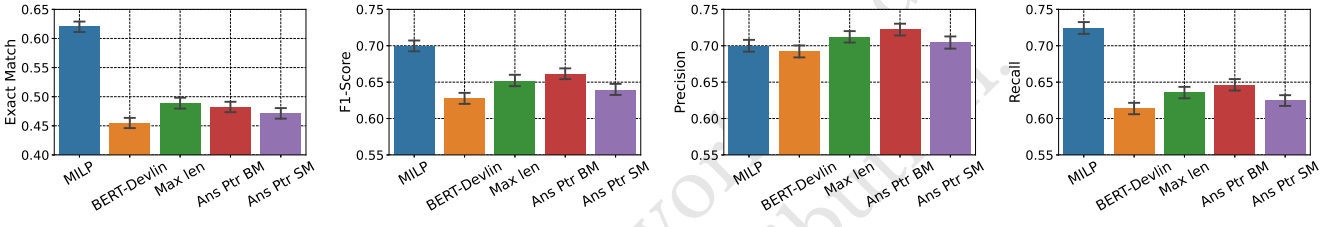


(a) Exact match. (b) F1-Score. (c) Precision. (d) Recall.

	p-value EM	*p*-value F1-Score	*p*-value Precision	*p*-value Recall
MILP vs. BERT-Devlin	0.0000	0.0000	0.0750	0.0000
MILP vs. Max len	0.0000	0.0000	0.1660	0.0000
MILP vs. Ans Ptr BM	0.0000	0.0000	0.5233	0.0000
MILP vs. Ans Ptr SM	0.0000	0.0000	0.0000	0.0000

(e) Statistical analysis: we call a ***statistically significant improvement*** when p-value≤ 0.05.

Figure 10: Performance comparison on the SQuAD1.1 dataset. 95% confidence interval is shown.

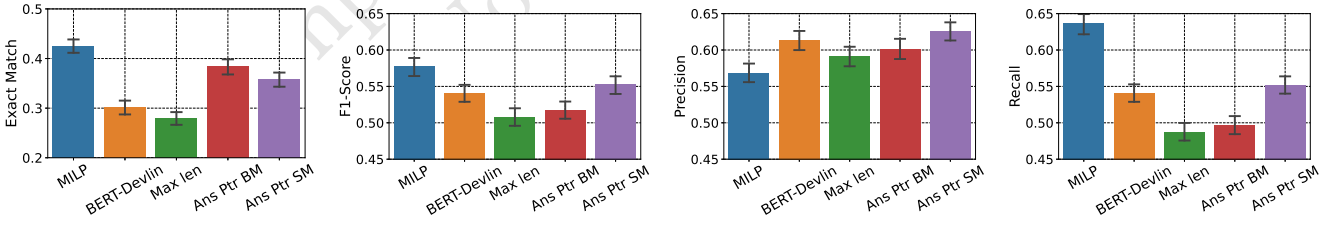


(a) Exact match. (b) F1-Score. (c) Precision. (d) Recall.

	p-value EM	*p*-value F1-Score	*p*-value Precision	*p*-value Recall
MILP vs. BERT-Devlin	0.0000	0.0000	0.2018	0.0000
MILP vs. Max len	0.0000	0.0000	0.0318	0.0000
MILP vs. Ans Ptr BM	0.0000	0.0000	0.0001	0.0000
MILP vs. Ans Ptr SM	0.0000	0.0000	0.4181	0.0000

(e) Statistical analysis: we call a ***statistically significant improvement*** when p-value≤ 0.05.

Figure 11: Performance comparison on the SQuAD2.0 dataset. 95% confidence interval is shown.



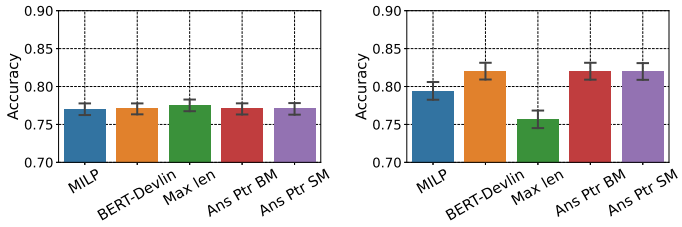
(a) Exact match. (b) F1-Score. (c) Precision. (d) Recall.

	p-value EM	*p*-value F1-Score	*p*-value Precision	*p*-value Recall
MILP vs. BERT-Devlin	0.0000	0.0001	0.0000	0.0000
MILP vs. Max len	0.0000	0.0000	0.0204	0.0000
MILP vs. Ans Ptr BM	0.0001	0.0000	0.0009	0.0000
MILP vs. Ans Ptr SM	0.0000	0.0000	0.0000	0.0000

(e) Statistical analysis: we call a ***statistically significant improvement*** when p-value≤ 0.05.

Figure 12: Performance comparison on the NewsQA dataset. 95% confidence interval is shown.

Unpublished working draft. Not for distribution.



(a) SQuAD2.0. (b) NewsQA.

Figure 13: Performance of the threshold-based answerable verification method.

- **Ans Ptr BM**: this method is the boundary model output depicted in Figure 1.b in [45]. Similarly, this method uses Pointer Net to only select two tokens from the input passage, and all the tokens between these two tokens in the passage are treated as the answer. This methods uses a TAV by averaging over $P(j)$ in a candidate answer to identify (un)answerable questions.

To summarize all the results obtained, Figures 10, 11, and 12 show the Exact Match, the F1-Score, the Precision and the Recall obtained on all datasets for our approach to which we refer as MILP, and the baselines with a statistical analysis of significance. From these results, we make the following **observations**:

- First, in terms of Exact Match, we observe in Figures 10a, 11a, and 12a that our method outperforms all baselines. Moreover, our statistical analysis shows that this improvement is statistically significant as p-value ≤ 0.05. This shows that the first step of our approach is efficient in maximizing Exact Match.
- Second, in terms of F1-Score, we observe in Figures 10b, 11b, and 12b that our approach achieves the best performance in all datasets, with a statistically significant improvement over all baselines (p-value ≤ 0.05). This is because our MILP-based approach seeks to maximize directly *(expected)* F1-Score.
- Third, we observe in Figures 10c, 11c, and 12c that achieves the worst performance in terms of Precision. This is due to the fact that all baselines seeks a substring with tokens that have a high probability of relevance, which increases the Precision at the expense of Recall. Moreover, we note that this improvement is statistically significant with almost all baselines, except on the NewsQA dataset.
- Fourth, in terms of recall, we observe in Figures 10d, 11d, and 12d that our method achieves the best performance while having a statistically significant improvement w.r.t. all Baselines – p-value ≤ 0.05.
- Finally, we show in Figure 13 the performance of the TAV based approaches. We note that our method achieves an accuracy of 80%, while being outperformed by all the baselines. However, we note that the improvement is statistically significant on the NewsQA dataset but not on the SQuAD2.0.

In sum, the empirical evaluation we have performed and the results we have obtained show that our MILP-based optimization approach for text spans extraction is efficient and competitive both in terms of performance and time complexity. During this last evaluation, the average execution time to answer a question was $0.604s \pm 0.039$, which shows that it can be used in practice. This is due to the fact that the Sigmoid activation function in the last classification layer of the neural network tends to output values that are either very close to 0 or to 1, which induces a low level of noise in the vector of relevance probability of tokens. Thus, the feasible region is reduced, which allows the optimization algorithm to scale well – see Figure 8.

6 CONCLUSION

In this paper, we have proposed a novel model agnostic approach for text spans extraction. The proposed double-edged sword method is a 2-step approach that consists of a simplified deep neural network architecture with a single classification output layer that aims to maximize exact-match, followed by an *independent* optimization module that extracts the answer text span by maximizing an *expected* F1-Score (EF1) objective. We derived the *expected* F1-Score as an objective criterion for text span extraction, and we have experimentally demonstrated that the proposed expected F1-Score metric is a good surrogate of the ground truth F1-Score. We have also proposed a method for optimization of expected F1-score using a MILP formulation subject to parameterized constraints for text spans extraction. The proposed approach allows the extraction of multiple possible answers to a question from a passage and to identify if a span of text is a valid answer and so the answerability of the question given the passage. Finally, we have demonstrated the effectiveness of our approach with extensive experimental evaluations on the SQuAD1.1, SQuAD2.0, and NewsQA datasets that showed statistically significant improvement comparatively. Future work includes exploring more efficient optimization algorithms for improving the time complexity of our proposed approach.

REFERENCES

[1] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas, November 2016. Association for Computational Linguistics.

[2] Adam Trischler, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordoni, Philip Bachman, and Kaheer Suleman. NewsQA: A machine comprehension dataset. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, pages 191–200, Vancouver, Canada, August 2017. Association for Computational Linguistics.

[3] Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don't know: Unanswerable questions for SQuAD. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 784–789, Melbourne, Australia, July 2018. Association for Computational Linguistics.

[4] Eunsol Choi, He He, Mohit Iyyer, Mark Yatskar, Wen-tau Yih, Yejin Choi, Percy Liang, and Luke Zettlemoyer. QuAC: Question answering in context. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2174–2184, Brussels, Belgium, October-November 2018. Association for Computational Linguistics.

[5] Siva Reddy, Danqi Chen, and Christopher D. Manning. CoQA: A conversational question answering challenge. *Transactions of the Association for Computational Linguistics*, 7:249–266, March 2019.

[6] Yongfeng Zhang, Xu Chen, Qingyao Ai, Liu Yang, and W. Bruce Croft. Towards conversational search and recommendation: System ask, user respond. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, CIKM '18, page 177–186, New York, NY, USA, 2018. Association for Computing Machinery.

[7] Dietmar Jannach, Ahtsham Manzoor, Wanling Cai, and Li Chen. A survey on conversational recommender systems. *arXiv preprint arXiv:2004.00646*, 2020.

[8] Lei Cui, Shaohan Huang, Furu Wei, Chuanqi Tan, Chaoqun Duan, and Ming Zhou. SuperAgent: A customer service chatbot for E-commerce websites. In *Proceedings of ACL 2017, System Demonstrations*, pages 97–102, Vancouver, Canada, July 2017. Association for Computational Linguistics.

Unpublished working draft. Not for distribution.

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

[10] Amit Mishra and Sanjay Kuma Jain. A survey on question answering systems with classification. *Journal of King Saud University - Computer and Information Sciences*, 28(3):345 – 361, 2016.

[11] Oleksandr Kolomiyets and Marie-Francine Moens. A survey on question answering technology from an information retrieval perspective. *Information Sciences*, 181(24):5412 – 5434, 2011.

[12] Bert F. Green, Alice K. Wolf, Carol Chomsky, and Kenneth Laughery. Baseball: An automatic question-answerer. In *Papers Presented at the May 9-11, 1961, Western Joint IRE-AIEE-ACM Computer Conference*, IRE-AIEE-ACM '61 (Western), page 219–224, New York, NY, USA, 1961. Association for Computing Machinery.

[13] W. A. Woods. Progress in natural language understanding: An application to lunar geology. In *Proceedings of the June 4-8, 1973, National Computer Conference and Exposition*, AFIPS '73, page 441–450, New York, NY, USA, 1973. Association for Computing Machinery.

[14] Ming Zhu, Aman Ahuja, Wei Wei, and Chandan K. Reddy. A hierarchical attention retrieval model for healthcare question answering. In *The World Wide Web Conference*, WWW '19, page 2472–2482, New York, NY, USA, 2019. Association for Computing Machinery.

[15] Feng Luo, Xiaoli Wang, Qingfeng Wu, Jiaying Liang, Xueliang Qiu, and Zhifeng Bao. Hqadeephelper: A deep learning system for healthcare question answering. In *Companion Proceedings of the Web Conference 2020*, WWW '20, page 194–197, New York, NY, USA, 2020. Association for Computing Machinery.

[16] Jeanne E. Daniel, Willie Brink, Ryan Eloff, and Charles Copley. Towards automating healthcare question answering in a noisy multilingual low-resource setting. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 948–953, Florence, Italy, July 2019. Association for Computational Linguistics.

[17] Biralatei Fawei, Jeff Z Pan, Martin Kollingbaum, and Adam Z Wyner. A semi-automated ontology construction for legal question answering. *New Generation Computing*, 37(4):453–478, 2019.

[18] Phong-Khac Do, Huy-Tien Nguyen, Chien-Xuan Tran, Minh-Tien Nguyen, and Minh-Le Nguyen. Legal question answering using ranking svm and deep convolutional neural network. *arXiv preprint arXiv:1703.05320*, 2017.

[19] Sweta P. Lende and M. M. Raghuwanshi. Question answering system on education acts using nlp techniques. In *2016 World Conference on Futuristic Trends in Research and Innovation for Social Welfare (Startup Conclave)*, pages 1–6, 2016.

[20] Caner Derici, Kerem Çelik, Ekrem Kutbay, Yiğit Aydın, Tunga Güngör, Arzucan Özgür, and Günizi Kartal. Question analysis for a closed domain question answering system. In Alexander Gelbukh, editor, *Computational Linguistics and Intelligent Text Processing*, pages 468–482, Cham, 2015. Springer International Publishing.

[21] David Ferrucci, Eric Brown, Jennifer Chu-Carroll, James Fan, David Gondek, Aditya A Kalyanpur, Adam Lally, J William Murdock, Eric Nyberg, John Prager, et al. Building watson: An overview of the deepqa project. *AI magazine*, 31(3):59–79, 2010.

[22] Mohit Iyyer, Wen-tau Yih, and Ming-Wei Chang. Search-based neural structured learning for sequential question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1821–1831, 2017.

[23] Haitian Sun, Bhuwan Dhingra, Manzil Zaheer, Kathryn Mazaitis, Ruslan Salakhutdinov, and William W Cohen. Open domain question answering using early fusion of knowledge bases and text. *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4231–4242 Brussels, Belgium, October 31 - November 4, 2018.*

[24] Ajitkumar M Pundge, SA Khillare, and C Namrata Mahender. Question answering system, approaches and techniques: A review. *International Journal of Computer Applications*, 141(3):0975–8887, 2016.

[25] Joseph Weizenbaum. Eliza—a computer program for the study of natural language communication between man and machine. *Commun. ACM*, 9(1):36–45, January 1966.

[26] Daniel G. Bobrow, Ronald M. Kaplan, Martin Kay, Donald A. Norman, Henry Thompson, and Terry Winograd. Gus, a frame-driven dialog system. *Artificial Intelligence*, 8(2):155–173, 1977.

[27] Jun Suzuki, Yutaka Sasaki, and Eisaku Maeda. SVM answer selection for open-domain question answering. In *COLING 2002: The 19th International Conference on Computational Linguistics*, 2002.

[28] Silvia Quarteroni and Suresh Manandhar. Designing an interactive open-domain question answering system. *Natural Language Engineering*, 15(1):73, 2009.

[29] Dongfeng Cai, Yanju Dong, Dexin Lv, Guiping Zhang, and Xuelei Miao. A web-based chinese question answering with answering validation. In *2005 International Conference on Natural Language Processing and Knowledge Engineering*, pages 499–502, 2005.

[30] Abraham Ittycheriah and Salim Roukos. Ibm's statistical question answering system-trec-11. Technical report, IBM THOMAS J WATSON RESEARCH CENTER YORKTOWN HEIGHTS NY, 2006.

[31] Sanjay K. Dwivedi and Vaishali Singh. Research and reviews in question answering system. *Procedia Technology*, 10:417–424, 2013. First International Conference on Computational Intelligence: Modeling Techniques and Applications (CIMTA) 2013.

[32] Deepak Ravichandran and Eduard Hovy. Learning surface text patterns for a question answering system. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, page 41–47, USA, 2002. Association for Computational Linguistics.

[33] Hang Cui, Min-Yen Kan, and Tat-Seng Chua. Soft pattern matching models for definitional question answering. *ACM Trans. Inf. Syst.*, 25(2):8–es, April 2007.

[34] Eriks Sneiders. Automated question answering using question templates that cover the conceptual model of the database. In Birger Andersson, Maria Bergholtz, and Paul Johannesson, editors, *Natural Language Processing and Information Systems*, pages 235–239, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg.

[35] Christina Unger, Lorenz Bühmann, Jens Lehmann, Axel-Cyrille Ngonga Ngomo, Daniel Gerber, and Philipp Cimiano. Template-based question answering over rdf data. In *Proceedings of the 21st International Conference on World Wide Web*, WWW '12, page 639–648, New York, NY, USA, 2012. Association for Computing Machinery.

[36] Zhuosheng Zhang, Yuwei Wu, Hai Zhao, Zuchao Li, Shuailiang Zhang, Xi Zhou, and Xiang Zhou. Semantics-aware bert for language understanding. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):9628–9635, Apr. 2020.

[37] Fu Sun, Linyang Li, Xipeng Qiu, and Yang Liu. U-net: Machine reading comprehension with unanswerable questions, 2018.

[38] Minghao Hu, Furu Wei, Yuxing Peng, Zhen Huang, Nan Yang, and Dongsheng Li. Read + verify: Machine reading comprehension with unanswerable questions. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):6529–6537, Jul. 2019.

[39] Xiaodong Liu, Yelong Shen, Kevin Duh, and Jianfeng Gao. Stochastic answer networks for machine reading comprehension. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1694–1704, Melbourne, Australia, July 2018. Association for Computational Linguistics.

[40] Hsin-Yuan Huang, Chenguang Zhu, Yelong Shen, and Weizhu Chen. Fusionnet: Fusing via fully-aware attention with application to machine comprehension. In *International Conference on Learning Representations*, 2018.

[41] Zhuosheng Zhang, Junjie Yang, and Hai Zhao. Retrospective reader for machine reading comprehension. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05), 2021.

[42] Zhuosheng Zhang, Yuwei Wu, Junru Zhou, Sufeng Duan, Hai Zhao, and Rui Wang. Sg-net: Syntax-guided machine reading comprehension. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):9636–9643, Apr. 2020.

[43] Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S. Weld, Luke Zettlemoyer, and Omer Levy. SpanBERT: Improving pre-training by representing and predicting spans. *Transactions of the Association for Computational Linguistics*, 8:64–77, 2020.

[44] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28, pages 2692–2700. Curran Associates, Inc., 2015.

[45] Shuohang Wang and Jing Jiang. Machine comprehension using match-lstm and answer pointer. In *International Conference on Learning Representations*, 2017.

[46] Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.

[47] G.M.P. van Kempen and L.J. van Vliet. Mean and variance of ratio estimators used in fluorescence ratio imaging. *Cytometry*, 39(4):300–305, 2000.

[48] Abraham Charnes and William W. Cooper. Programming with linear fractional functionals. *Naval Research Logistics Quarterly*, 9(3-4):181–186, 1962.

[49] Fred Glover. Improved linear integer programming formulations of nonlinear integer problems. *Management Science*, 22(4):455–460, 1975.

[50] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[51] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[52] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations. In *International Conference on Learning Representations*,

Unpublished working draft. Not for distribution.

2020.

Unpublished working draft.
Not for distribution.