A Simple and Effective Model for Answering Multi-span Questions

Elad Segal[1], Avia Efrat[1], Mor Shoham[1], Amir Globerson[1,3], Jonathan Berant[1,2]

[1]Tel Aviv University [2]Allen Institute for AI [3]Google Research
elad.segal@gmail.com, {aviaefra,morshoham}@mail.tau.ac.il
amir.globerson@gmail.com, joberant@cs.tau.ac.il

Abstract

Models for reading comprehension (RC) commonly restrict their output space to the set of all single contiguous spans from the input, in order to alleviate the learning problem and avoid the need for a model that generates text explicitly. However, forcing an answer to be a single span can be restrictive, and some recent datasets also include *multi-span questions*, i.e., questions whose answer is a set of non-contiguous spans in the text. Naturally, models that return single spans cannot answer these questions. In this work, we propose a simple architecture for answering multi-span questions by casting the task as a sequence tagging problem, namely, predicting for each input token whether it should be part of the output or not. Our model substantially improves performance on span extraction questions from DROP and QUOREF by 9.9 and 5.5 EM points respectively.

1 Introduction

The task of reading comprehension (RC), where given a question and context, one provides an answer, has gained immense attention recently. In most datasets and models (Rajpurkar et al., 2016; Trischler et al., 2016; Seo et al., 2017; Yu et al., 2018; Kwiatkowski et al., 2019), RC is set up as an *extractive* task, where the answer is constrained to be a single span from the input. This makes learning easier, since the model does not need to generate text *abstractively*, while still being expressive enough to capture a large set of questions.

However, for some questions, while the answer is indeed extractive, i.e., contained in the input, it is not *a single span*. For example, in Figure 1 the answer includes two people who appear as non-contiguous spans in the context. Existing models (Seo et al., 2017; Dua et al., 2019) are by design unable to provide the correct answer to such *multi-span questions*.

Figure 1: A *multi-span question* from DROP, and a `BIO` tagging for it (`O` tags omitted). The first occurrence of *Barack Obama* does not answer the question.

While most work has largely ignored this issue, recent work has taken initial steps towards handling multi-span questions. Hu et al. (2019) proposed to predict the number of output spans for each question, and used a non-differentiable inference procedure to find them in the text, leading to a complex training procedure. Andor et al. (2019) proposed a `Merge` operation that merges spans, but is constrained to at most 2 spans. Chen et al. (2020) proposed a non-differentiable symbolic approach which outputs programs that compose single-span extractions.

In this work, we propose a simple and fully differentiable architecture for handling multi-span questions that evades the aforementioned shortcomings, and outperforms prior work. Similar to Yao et al. (2013), who used a linear model over tree-based features, we cast question answering as a *sequence tagging task*, predicting for each token whether it is part of the answer. At test time, we decode the answer with standard decoding methods, such as Viterbi.

We show the efficacy of our approach on span-extraction questions from both the DROP (Dua et al., 2019) and QUOREF (Dasigi et al., 2019) datasets. Replacing the single-span architecture with our multi-span approach improves performance by 7.8 and 5.5 EM points respectively. Com-

bining the single-span and multi-span architectures further improves performance by 2.1 EM on DROP, surpassing results by other span-extraction methods on both datasets.

2 Background: Single-span Model

Setup Given a training set of question-context-answer triplets $(q_i, c_i, a_i)_{i=1}^N$, our goal is to learn a function that maps a question-context pair (q, c) to an answer a. We briefly review the standard *single-span architecture* for RC (Devlin et al., 2019), which we build upon.

First, we encode the question and context with a pre-trained language model, such as BERT (Devlin et al., 2019): $\boldsymbol{h} = \text{Encoder}([q, c])$, where $\boldsymbol{h} = (\boldsymbol{h}_1, \ldots, \boldsymbol{h}_m)$ is a sequence of contextualized representations for all input tokens. Then, two parameterized functions (feed-forward networks), $f_{\text{start}}(\boldsymbol{h}_i)$ and $f_{\text{end}}(\boldsymbol{h}_i)$, are used to compute a score for each token, corresponding to whether that token is the start or the end of the answer. Last, the start and end probability for each token i is computed as follows:

$$\boldsymbol{p}_i^{\text{start}} = \text{softmax}\left(f_{\text{start}}(\boldsymbol{h}_1), \ldots, f_{\text{start}}(\boldsymbol{h}_m)\right)_i,$$

$$\boldsymbol{p}_i^{\text{end}} = \text{softmax}\left(f_{\text{end}}(\boldsymbol{h}_1), \ldots, f_{\text{end}}(\boldsymbol{h}_m)\right)_i,$$

where both $\boldsymbol{p}^{\text{start}}, \boldsymbol{p}^{\text{end}} \in \mathbb{R}^{m \times 1}$. Training is done by minimizing cross entropy of the start and end indices of the gold span, and at test time the answer span is extracted by finding the indices (s, e):

$$(s, e) = \arg\max_{s \leq e} \ \boldsymbol{p}_s^{\text{start}} \boldsymbol{p}_e^{\text{end}}.$$

3 Multi-span Model

3.1 Span Extraction as Sequence Tagging

Extracting a variable number of spans from an input text is standard in many natural language processing tasks, such as Named Entity Recognition (NER) and is commonly cast as a sequence tagging problem (Ramshaw and Marcus, 1995). Here we apply this approach to multi-span questions.

Our model uses the same contextualized representations $\boldsymbol{h}$, but rather than predicting start and end probabilities, it outputs a probability distribution over a set of tags for each token. We experiment with two tagging schemes. First, the well-known BIO tagging (Sang, 2000; Huang et al., 2015), in which B denotes the first token of an output span, I denotes subsequent tokens in a span, and O denotes tokens that are not part of an output span. In addition, we experiment with a simpler IO

tagging scheme, where words are tagged as either part of the answer (I) or not (O). Formally, given a tagging scheme with $|S|$ tags ($|S| = 3$ for BIO and $|S| = 2$ for IO), for each of the m tokens, the probability for the tag of the i-th token is

$$\boldsymbol{p}_i = \text{softmax}(f(\boldsymbol{h}_i)) \qquad (1)$$

where $\boldsymbol{p} \in \mathbb{R}^{m \times |S|}$, and f is a parameterized function with $|S|$ outputs.

3.2 Training

Assume each answer a is a set of strings, where each string corresponds to a span in the input. We would like to train our model to predict the correct output for this set of spans. When the answer spans appear only once in the input, this is simple, since the ground-truth tagging is immediately available. However, there are many cases where a given answer span appears multiple times in the input. We next explain how to address this.

To illustrate, consider the following simple example (assume a BIO scheme). Given the input "X Y Z Y Z" and the correct multi-span answer {"X", "Z"}, there are three possible gold taggings: B O B O B, B O B O O, and B O O O B. Thus, the ground-truth BIO cannot be determined unambiguously in this case. Figure 1 illustrates this issue with a real example from DROP.[1]

To tackle the above issue, we enumerate over the set of all *possibly-correct taggings*, $\mathcal{T}$, where given a multi-span answer a, a possibly-correct tagging is one in which all gold answer spans are tagged as such at least once.[2] We train our models by maximizing the marginal probability of all possibly-correct taggings:

$$\log p(\mathcal{T} \mid \boldsymbol{h}) = \log \sum_{\boldsymbol{T} \in \mathcal{T}} \left(\prod_{i=1}^m \boldsymbol{p}_i[\boldsymbol{T}_i] \right),$$

where $\boldsymbol{p}_i[\boldsymbol{T}_i]$ (see Eq. (1)) is the probability the model assigns to token i having the tag $\boldsymbol{T}_i$. The loss is minimized when $\boldsymbol{p}$ gives probability 1.0 to one of the possibly-correct taggings in $\mathcal{T}$.

3.3 Decoding Spans from a Tagging

At test time, given predicted tag probabilities $\boldsymbol{p}$, we would like to find the most likely tagging $\hat{\boldsymbol{T}}$. Let $\mathcal{V}$

---

[1] In QUOREF, the indices of the gold answer spans are explicitly given, so a single gold tagging can be defined.

[2] While $|\mathcal{T}|$ can grow exponentially with the number of spans in an answer, in practice $|\mathcal{T}|$ is at most 1000 for 99.66% of the examples of DROP, and so we can enumerate over $\mathcal{T}$ directly in these cases. In the other 0.34%, we take a single tagging that marks all occurrences of the answer spans.

be the set of all valid taggings. We wish to find:

$$\hat{\boldsymbol{T}} = \arg\max_{\boldsymbol{T} \in \mathcal{V}} \prod_{i=1}^{m} \boldsymbol{p}_i[\boldsymbol{T}_i].$$

For `BIO` tags, the set $\mathcal{V}$ comprises all taggings that don't include an `I` after an `O`, and the maximization problem can be solved in linear time using Viterbi decoding (Viterbi, 1967) as in Yao et al. (2013); Mehta et al. (2018). For `IO` tags, all taggings are valid, and maximization is done by predicting the tag with highest probability in each token independently. Because answer spans are (practically) never adjacent in RC, an `IO`-tagging produces a set of spans by choosing all maximal spans that are contiguously tagged with `I`.

4 "Multi-Head" Models

Some RC datasets contain questions where the output is not necessarily a span. For example, in DROP, the answer to some questions is a number that is not in the text, but can be computed by performing arithmetic operations. To handle such cases, many models (Dua et al., 2019; Hu et al., 2019) employ a *multi-head architecture*. In these models, each *head z* is a small module that takes the contextualized representations $\boldsymbol{h}$ as input and computes a probability distribution over answers $p_z(a \mid q, c) = p_z(a \mid \boldsymbol{h})$. For example, in Hu et al. (2019), there are two heads that output spans, and three heads that output numbers. To determine which head to use for each question, an additional module is trained: $p_{\text{head}}(z \mid q, c) = p_{\text{head}}(z \mid \boldsymbol{h})$. Thus, the model probability for an answer is:

$$p(a \mid q, c) = \sum_z p_{\text{head}}(z \mid q, c) \cdot p_z(a \mid q, c).$$

With this architecture, we can seamlessly integrate our multi-span approach into existing RC models. Specifically, a model can include both a single-span head and a multi-span head, dynamically deciding which span extraction method to utilize based on the input.

5 Empirical Evaluation

Experimental setup As an encoder, we use the Hugging Face implementation of RoBERTa$_{\text{LARGE}}$ (Wolf et al., 2019; Liu et al., 2019), which produces the representations $\boldsymbol{h}$. For DROP, we add the arithmetic and count heads from Dua et al. (2019) to handle non-span questions. Full details of the experimental setup are in Appendix A.

5.1 Results

Table 1 shows development set results on the *span-extraction questions* of DROP (Dua et al., 2019) and QUOREF (Dasigi et al., 2019). We compare the previous best-performing multi-span models to a combination of our multi-span architecture (TASE: TAg-based Span Extraction) with the traditional single-span extraction (SSE), as well as to each separately.

Comparison to previous models For a fair comparison with prior work on DROP, we also train our model initialized with BERT$_{\text{LARGE}}$, as all prior work used it as an encoder. On DROP, TASE$_{\text{BIO}}$+SSE (BERT$_{\text{LARGE}}$) outperforms all prior models that handle multi-span questions, improving by at least 3.2 EM points. On multi-span questions, we dramatically improve performance over BERT-CALC and MTMSN, while obtaining similar performance to NeRd. On QUOREF, compared to CorefRoBERTa$_{\text{LARGE}}$ (Ye et al., 2020) which uses the same method as MTMSN for multi-span extraction, we achieve a substantial improvement of over 20 EM on multi-span questions and an improvement of 4.5 EM and 3.2 F$_1$ on the full development set, where the best results are achieved when using solely our multi-span architecture with `IO`-tagging.

Comparing span extraction architectures Table 1 also shows that in both DROP and QUOREF, *replacing* the single-span extraction architecture with our multi-span extraction results in dramatic improvement in multi-span question performance, while single-span question performance is either maintained or improved. Furthermore, although combining both architectures tends to yield the best overall performance,[3] the improvement over using only our multi-span architecture is not substantial, suggesting that the multi-span architecture may be used by itself as a general span extraction method.

Effects of tagging scheme Overall, the results are quite similar for the `BIO` and `IO` schemes. The slight advantage of `IO` could perhaps be explained by the fact that the model no longer requires distinguishing between `B` and `I`, in the presence of powerful contextualized representations.

---

[3]As single-span questions outnumber multi-span questions in DROP and QUOREF 1:7 and 1:10 respectively, the overall span performance ("All Spans") gives a much larger weight to single-span performance.

Model	DROP						QUOREF					
	All Spans		Multi-Span		Single-Span		All Spans		Multi-Span		Single-Span	
	EM	F1	EM	F1	EM	F1	EM	F1	EM	F1	EM	F1
BERT-CALC	69.1	78.9	6.2	47.0	79.8	84.3	-	-	-	-	-	-
MTMSN	69.7	79.9	25.1	62.8	77.5	82.8	-	-	-	-	-	-
NeRd	73.2	81.3	51.3	77.6	76.2	81.8	-	-	-	-	-	-
CorefRoBERTa$_{LARGE}$	-	-	-	-	-	-	74.9	81.7	38.8	65.9	78.7	83.3
TASE$_{BIO}$ + SSE (BERT$_{LARGE}$)	76.4	83.9	53.6	76.9	80.2	85.1	75.8	81.1	52.5	76.7	78.2	81.6
TASE$_{BIO}$ + SSE	79.7	87.1	56.3	79.9	83.6	88.3	79.0	84.2	**59.7**	**80.0**	80.9	84.6
TASE$_{IO}$ + SSE	**80.5**	**87.8**	**58.5**	**80.7**	**84.2**	**89.0**	79.4	84.8	57.9	79.2	**81.6**	**85.4**
TASE$_{BIO}$	77.9	85.5	56.6	79.3	81.5	86.6	78.9	84.6	56.6	77.5	81.2	85.3
TASE$_{IO}$	78.4	86.8	56.8	79.8	82.1	88.0	**79.4**	**84.9**	59.3	**80.0**	81.4	**85.4**
SSE	70.6	80.2	0.0	37.6	81.5	86.7	73.9	80.7	0.0	37.4	81.4	85.0
TASE$_{BIO}$, NOMARG.	76.2	85.0	54.7	79.0	79.8	86.1	-	-	-	-	-	-

Table 1: Development set results on DROP and QUOREF questions whose answer is a span (or list of spans).
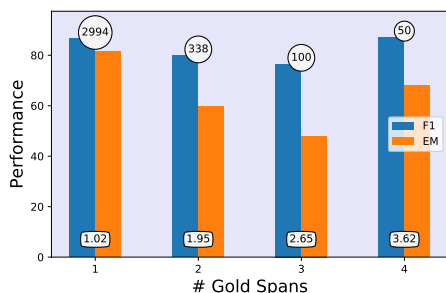

Figure 2: DROP Performance of TASE$_{BIO}$ by number of spans in the gold answer. Labels at the bottom indicate the average number of predicted spans. Circles at the top are the number of examples. These same trends are observed in QUOREF as well.

Effect of marginalization To check whether marginalizing over all possibly-correct taggings is beneficial, we ran TASE$_{BIO}$ in a setup where only a single tagging is considered, namely where all occurrences of a gold answer span are tagged. Table 1 shows that this indeed leads to a moderate drop of up to 2 points in performance.

Test set results We ran TASE$_{IO}$ on the QUOREF test set. Our model obtains 79.7 EM and 86.1 F1, an improvement of 3.9 EM points and 3.3 F1 points over the state-of-the-art CorefRoBERTa$_{LARGE}$. On DROP, our TASE$_{IO}$+SSE model achieves 80.4 EM and 83.6 F1 on the entire test set (including non-span questions).

We note that the top 10 models on the DROP leaderboard (as of September 15, 2020) have all incorporated our multi-span head using our code base which has been public for a while.

5.2 Analysis

Figure 2 shows that in both DROP and QUOREF the performance of TASE$_{BIO}$ decreases only moder-

ately as the number of gold spans increases. This shows relative robustness to the number of answer spans. In addition, we can see that our architecture is quite accurate in predicting the correct number of spans, with a tendency for under-estimation.

We analyzed the performance of the p_{head} module in TASE$_{BIO}$+SSE. A non-multi-span head is selected erroneously for 3.7% and 7.2% of the multi-span questions in DROP and QUOREF respectively. The multi-span head is selected for 1.2% and 1.5% of the single-span questions in DROP and QUOREF respectively. However, this is reasonable as the multi-span head is capable of answering single-span questions as well, and indeed it returned a single span in 45% of these cases on both datasets.

We manually analyzed errors of TASE$_{BIO}$+SSE on DROP, and detected 3 main failure cases: (1) questions where the answer is a span, but requires some numerical computation internally, (2) questions where the number of output spans is explicitly mentioned in the question but is not followed by the model, and (3) questions where a single contiguous span is unnecessarily split into two shorter spans. An example for each case is given in Appendix B.

6 Conclusion

In this work, we cast the task of answering multi-span questions as a sequence tagging problem, and present a simple corresponding multi-span architecture. We show that replacing the standard single-span architecture with our multi-span architecture dramatically improves results on multi-span questions, without harming performance on single-span questions, leading to state-of-the-art results on QUOREF. In addition, integrating our multi-span architecture into existing models further improves performance on DROP, as is evident from the leading models on

DROP's leaderboard. Our code can be downloaded from https://github.com/eladsegal/tag-based-multi-span-extraction.

References

Daniel Andor, Luheng He, Kenton Lee, and Emily Pitler. 2019. Giving BERT a calculator: Finding operations and arguments with reading comprehension. *EMNLP-IJCNLP*.

Xinyun Chen, Chen Liang, Adams Wei Yu, Denny Zhou, Dawn Song, and Quoc V Le. 2020. Neural symbolic reader: Scalable integration of distributed and symbolic representations for reading comprehension. In *ICLR*.

Pradeep Dasigi, Nelson F. Liu, Ana Marasović, Noah A. Smith, and Matt Gardner. 2019. Quoref: A reading comprehension dataset with questions requiring coreferential reasoning. In *Proc. of EMNLP-IJCNLP*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. 2019. DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs. In *Proc. of NAACL*.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke S. Zettlemoyer. 2017. AllenNLP: A deep semantic natural language processing platform.

Minghao Hu, Yuxing Peng, Zhen Huang, and Dongsheng Li. 2019. A multi-type multi-span network for reading comprehension that requires discrete reasoning. In *Proceedings of EMNLP*.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF models for sequence tagging.

Jambay Kinley and Raymond Lin. 2019. NABERT+: Improving numerical reasoning in reading comprehension. URL https://github.com/raylin1000/drop-bert.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. 2019. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466.

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

Sanket Vaibhav Mehta, Jay Yoon Lee, and Jaime G. Carbonell. 2018. Towards semi-supervised learning for deep semantic role labeling. In *EMNLP*.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *EMNLP*.

Lance Ramshaw and Mitch Marcus. 1995. Text chunking using transformation-based learning. In *Third Workshop on Very Large Corpora*.

Erik F. Tjong Kim Sang. 2000. Transforming a chunker to a parser. In *CLIN*.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Bidirectional attention flow for machine comprehension. In *ICLR*.

Adam Trischler, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordoni, Philip Bachman, and Kaheer Suleman. 2016. Newsqa: A machine comprehension dataset. *Workshop on Representation Learning for NLP*.

Andrew J. Viterbi. 1967. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Trans. Information Theory*, 13(2):260–269.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, R'emi Louf, Morgan Funtowicz, and Jamie Brew. 2019. HuggingFace's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771.

Xuchen Yao, Benjamin Van Durme, Chris Callison-Burch, and Peter Clark. 2013. Answer extraction as sequence tagging with tree edit distance. In *HLT-NAACL*.

Deming Ye, Yankai Lin, Jiaju Du, Zhenghao Liu, Maosong Sun, and Zhiyuan Liu. 2020. Coreferential reasoning learning for language representation. *ArXiv*.

Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V. Le. 2018. QANet: Combining local convolution with global self-attention for reading comprehension. In *ICLR*.

Appendix for "A Simple and Effective Model for Answering Multi-span Questions"

A Experimental Setup

We experiment with model variations that use either SSE, TASE, or their combination as a multi-head model. For DROP, we additionally use *arithmetic* and *count* heads based on (Dua et al., 2019; Kinley and Lin, 2019). Our model is implemented with PyTorch (Paszke et al., 2019) and AllenNLP (Gardner et al., 2017). For f in Eq. (1) we use a 2-layer feed-forward network with ReLU activations and $|S|$ outputs. We use the Hugging Face implementation of RoBERTa$_{\text{LARGE}}$ (Wolf et al., 2019; Liu et al., 2019) as the encoder in our model. 5% of DROP and 30% of QUOREF are inputs with over 512 tokens. Due to RoBERTa$_{\text{LARGE}}$'s limitation of 512 positional embeddings, we truncate inputs by removing over-flowing tokens from the passage, both at train and test time. We discard 3.87% of the training examples of DROP and 5.05% of the training example of QUOREF, which are cases when the answer cannot be outputted by the model (due to a dataset error, or truncation of the correct answer span). For training, the BertAdam[4] optimizer is used with default parameters and learning rates of either 5×10^{-6} or 10^{-5}. Hyperparameter search was not performed. We train on a single NVIDIA Titan XP with a batch size of 2 and gradient accumulation of 6, resulting in an effective batch size of 12, for 20 epochs with an early-stopping patience of 10. The average runtime per epoch is 3.5 hours. Evaluation was performed with the official evaluation scripts of

DROP and QUOREF. Our full implementation can be found at https://github.com/eladsegal/tag-based-multi-span-extraction.

B Failure Cases Examples

Table 2 contains example failure cases of TASE$_{\text{BIO}}$+SSE on DROP.

---

[4] https://github.com/huggingface/transformers/blob/694e2117f33d752ae89542e70b84533c52cb9142/README.md#optimizers

Question	Excerpt from Context	Gold Answer	Prediction
Which two nationalities have the same number of immigrants in Bahrain?	Indians, 125,000 Bangladeshis, 45,000 Pakistanis, 45,000 Filipinos, and 8,000 Indonesians	{"Filipinos", "Pakistanis"}	{"Filipinos", "Pakistanis", "Indonesians"}
What event happened first, Spain losing all territories it had gained since 1909, or the Spanish retaking their major fort at Monte Arruit?	August 1921, Spain lost all the territories it had gained since 1909 [...] By January 1922 the Spanish had retaken their major fort at Monte Arruit	{"Spain lost all the territories it had gained since 1909"}	{"August 1921, Spain lost all the", "territories it had gained since 1909"}

Table 2: Example failure cases of TASE_BIO+SSE on DROP. The first answer exhibits a lack of numeric reasoning and ignores the expected number of spans stated in the question. The second splits a correct span into two spans.