Unveiling Wisdom: Inspirational Quote Extraction using a Retrieval Augmented Multi-Task Reader

Anonymous ACL submission

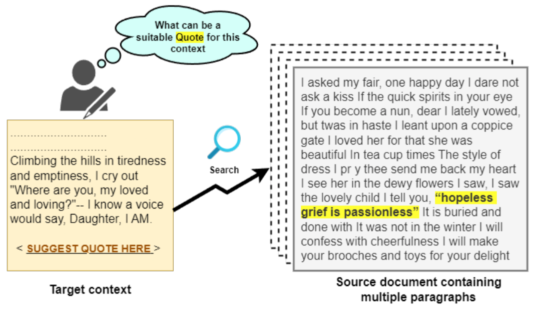
Abstract

Inspirational quotes from famous individuals are often used to convey thoughts in news articles, essays, and everyday conversations. In this paper, we propose a novel context-based quote extraction system that aims to predict the most relevant quote from a long text. We formulate this quote extraction as an open domain question answering problem first by employing a vector-store based retriever and then applying a multi-task reader. We curate three context-based quote extraction dataset and introduce a novel multi-task framework that improves the state-of-the-art performance, achieving a maximum improvement of 5.08% in BoW F1-score.

1 Introduction

Inspirational quotes from famous individuals are powerful tools that convey wisdom and insight in a concise and often figurative manner. They provide a secondary voice that reinforces the author's thoughts and beliefs (Liu et al., 2019). Context-aware quote extraction (also known as quote recommendation) is crucial in writing news articles, blogs, and summaries, as it helps to strengthen the expressed ideas. This process involves identifying phrases or sentences within a paragraph that are quotable and determining their relevance and quotability in a given context. Since "context" can be highly subjective, finding the most relevant quotes can be challenging due to the linguistic nuances involved. Figure 1 demonstrates a recommendation for a quotable phrase from a source paragraph, based on one context from the example of our dataset. It turns out that authors have to spend far too much time deciding what-to-quote from many source texts analyzing their context. Accordingly, it is in significant demand to automate the process of extracting quotes from a text.

To tackle the task, Bendersky and Smith (2012) attempts to identify "quotable" phrase from books


What can be a suitable Quote for this context
Search
Target context
Source document containing multiple paragraphs

Climbing the hills in tiredness and emptiness, I cry out "Where are you, my loved and loving?"-- I know a voice would say, Daughter, I AM.

< SUGGEST QUOTE HERE >

I asked my fair, one happy day I dare not ask a kiss If the quick spirits in your eye If you become a nun, dear I lately vowed, but twas in haste I leant upon a coppice gate I loved her for that she was beautiful In tea cup times The style of dress I pr y thee send me back my heart I see her in the dewy flowers I saw, I saw the lovely child I tell you, **"hopeless grief is passionless"** It is buried and done with It was not in the winter I will confess with cheerfulness I will make your brooches and toys for your delight

Figure 1: Example use-case of context-aware quote extraction from source document while composing an article. The highlighted portion from the source document can be a suitable quote for the target context in the left.

on the basis of linguistic and rhetorical properties. Unlike this, (Tan et al. (2015), Tan et al. (2016, 2018), Qi et al. (2022)), leverage "context" to select the most relevant quote from a list of quotes. (Lee et al. (2016); Wang et al. (2021)) use dialogue history as the *context*. The task of finding the most relevant quote itself remains challenging. Moreover, our task poses inherent difficulty, as we not only attempt to find the most relevant quote for a given context, but also extract the quote from a full source document (containing several hundreds of paragraphs). To the best of our knowledge, only (MacLaughlin et al., 2021) attempts to extract context-aware quotes from text documents (US presidential speech transcripts). However, the length of the documents are considerably small (see Table 1 for details) and they only cover the political domain for quote extraction. In addition, none of the experimental dataset apart from Qi et al. (2022) is publicly available.

In this research, we focus on bridging the gap by rigorously curating three datasets for context aware quote extraction task, and presenting a novel framework that can enhance the task of extracting quotes from a much longer text.

Our contributions are as follows.

- To better extract quotes based on the context, we propose a Retrieval Augmented Multi-task Reader (RA-MTR) framework that utilizes a vector-store for initial retrieval followed by *Llama-3* based re-ranker, and a multi-task framework that leverages two training tasks tailored specifically for the quote extraction scenario. In addition to the normal span prediction as suggested by (MacLaughlin et al., 2021), we incorporate quotability identification as a sequence classification task.

- We conduct rigorous experiments using RA-MTR and show that our framework outperforms the best-performing baseline by a maximum of **5.08%** in terms of BoW F1-score while considering the top-ranked paragraph as the location of the quote.
- We also perform analysis with the *multi-task reader* to demonstrate that our fine-tuned multi-task framework based on SpanBERT $\oplus$ SpanBERT-CRF improves the quote predictions over a series of baselines. Specifically, we train this multi-task framework using the popular QuoteR dataset (where the majority of the quotations are from books) (Qi et al., 2022) and evaluate on two different datasets – Quotus (which comprises political speech quotations) (Niculae et al., 2015) and 'Gandhi' (which is based on the quotes of Mahatma Gandhi)[1]. Our multi-task framework outperforms the standard BERT-based models by a large margin in a few shot settings. In particular, we see that even with eight data points from the target domain, our model beats BERT and SpanBERT by **14%** and **11%** in BoW F1-score respectively. The results are listed in Table 5.

2 Related work

We present this section subdivided into multiple important parts.

Quotability identification: (Bendersky and Smith, 2012) developed a quotable phrase extraction process that includes a supervised quotable phrase detection using lexical and syntactic features. (Wang et al., 2021) introduced a transformation matrix that directly maps the query representations to quotation representations. (Koto et al., 2014) gathered and examined memorable spoken quotes from TED public speaking based on speech duration, the fundamental frequency of speech signal (F0), and popularity.

Context aware quote recommendation: (Tan et al., 2015) proposed a learning-to-rank framework. (Tan et al., 2016) proposed a quote recommendation framework by learning the distributed meaning representations for the contexts and the quotes using LSTM. (Lee et al., 2016) built a quote recommender system to predict quotes based on Twitter dialogues as context. (Qi et al., 2022) built a large and the first publicly available dataset of quote recommendation.

Quotable paragraph identification: (MacLaughlin and Smith, 2021) utilized BERT-based models for ranking the quotable paragraphs evaluating on five different datasets. (Voskarides et al., 2021) discussed challenges of retrieving news articles in the context of developing event-centric narratives.

Quote identification and recommendation: To the best of our knowledge (MacLaughlin et al., 2021) made the first attempt to simultaneously rank the most quotable paragraphs and predict the most quotable spans from source transcripts modeling quote recommendation as an open-QA problem.

The present work: We extend the work of (MacLaughlin et al., 2021), by proposing a novel retriever augmented multi-task reader based quote extraction. The framework employs a vector-store based paragraph retriever followed by a decoder-only transformers based re-ranker and a novel multi-task based reader containing a sequence classification module for identifying quotable phrases along with context aware span prediction. We curate three datasets of different genres and evaluate our approach. Our method outperforms all the previous baselines and generalizes better in a cross-domain few-shot setting.

3 Approach

We formalize the problem as an open-QA task, similar to the one described in (MacLaughlin et al., 2021). Given a target context (T_C), and a source document (S_D) which consists of several paragraphs ($= \{P_1, P_2, .., P_n\}$), we require to first identify the most relevant list of paragraphs depending upon T_C, and then identify the most quotable phrase from the selected paragraphs. We propose the overall quote extraction approach consisting of a *retriever* (detailed in section 5.1.1) to select the most relevant paragraph followed by a *multi-task reader* (detailed in section 5.1) to extract a quote.

[1] https://www.goodreads.com/author/quotes/5810891.Mahatma_Gandhi

4 Dataset curation

In this section we present the details of the datasets first by discussing the quotes that we consider, followed by construction of *source paragraph* and *target context* for our experiments.

4.1 Training data

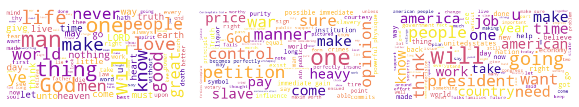
QuoteR: We primarily consider the English subset of the QuoteR dataset proposed by (Qi et al., 2022), known to be the largest publicly available dataset for the quote recommendation task. The authors of the corresponding paper collected several quotes from the popular *WikiQuote*[2] project and search for the occurrences of these quotes in the Project Gutenberg corpus[3], the BookCorpus (Zhu et al., 2015), and the OpenWebText corpus (Gokaslan and Cohen, 2019) respectively, and considered the preceding and the following 40 words of a particular quote to its left and right context respectively. After preprocessing, the authors provided a total of **6108** unique quotes and around **127k** contexts for those quotes.

4.2 Test data

Gandhi quotes: Websites such as mkgandhi[4] has made the Collected Works of Mahatma Gandhi (CWMG) publicly available, which is a huge text corpus consisting of 100 volumes. We collect around a total of 800 Mahatma Gandhi quotes in English from Goodreads[5] and the *mkgandhi* portal.
Quotus data: The authors in (MacLaughlin et al., 2021) utilizes the *Quotus* (Niculae et al., 2015) dataset for their experiments. The dataset consists of two sets of texts — transcripts of US Presidential speeches and press conferences from 2009-2014 (*the source document*), and news articles that report on the speeches (*the target document*). The authors crawled the articles and transcripts from the provided links in the Quotus data, and preprocessed them to gather a significant amount of quote, contexts, and paragraphs. However, they did not make their dataset publicly available. We ourselves re-scraped the links from the source Quotus data.
Curating source paragraph and target context: From these three dataset (i.e., *QuoteR*, *Gandhi*, and *Quotus*) we get a list of quotes. However, to evaluate our *retriever* and *reader*, we require to curate the source paragraph and the target context for each of the quotes. We leverage the *Project Gutenberg* corpus to construct 4,889 quote-context-paragraph triples (containing 1,708 unique quotes) for QuoteR. We use *Gadhipedia*[6] search engine to curate 737 triples for the Gandhi quotes. For the Quotus, we utilize the *Quoting POTUS*[7] website containing the news articles and align the quotes to source transcript for constructing 2,698 triples. The detailed steps and algorithms are provided in Appendix D.

4.3 Dataset statistics

Thus, overall we consider *three* datasets each from a different genre - (i) QuoteR - where most of the quotes are from novels, 2) Gandhi - solely based on the quotes of Mahatma Gandhi, and 3) Quotus - quotes from the political speech. The basic statistics of these three datasets are noted in Table 1. Figure 2 demonstrates the most prominent words present in the three datasets. The quotes in the QuoteR and Gandhi datasets contain positive words like "God", "good", "love", "truth", "petition" etc. The Quotus dataset on the other hand contains quotes having words "america", "president" etc. We also compare our dataset with the dataset used in other similar works (see Table 2). We present the only dataset containing quote, context and source paragraph. These datasets will be made publicly available upon acceptance.



(a) QuoteR (b) Gandhi (c) Quotus

Figure 2: Most prominent words present in the quotes across the three datasets.

5 Methodology

In this section, we describe the details of our methodology for quote extraction. We propose the overall quote extraction approach as an open-QA framework, which normally consists of a *retriever* and a *reader*. The retriever is essential for selecting the top paragraphs relevant to the context from the whole document. We employ a novel multi-task learning framework in the reader, which extracts the most quotable spans from the selected paragraphs and is discussed in detail in section 5.1. The

[2]https://en.wikiquote.org/
[3]https://www.gutenberg.org/
[4]https://mkgandhi.org/
[5]https://www.goodreads.com/author/quotes/5810891.Mahatma_Gandhi?page=35
[6]https://www.gandhipedia150.in
[7]http://snap.stanford.edu/quotus/vis/

Dataset	# of unique quotes	# of quote-context-paragraph triples	Avg. # of tokens / quote	Median # of tokens / quote	Avg # of para / Src	Avg # of tokens / Src
QuoteR	1708	4889	13.51	11	551	98783.17
Gandhi	737	737	20.42	19	19.54	3812.47
Quotus	2698	2698	20.46	16	86.78	4631.55

Table 1: Statistics for the three datasets. For QuoteR we report the instances that we could find in the Gutenberg corpus.

Dataset	Context based	Context type	Source paragraph	Public
Bendersky and Smith (2012)	✗	writings	✗	✗
Tan et al. (2015)	✓	writings	✗	✗
Wang et al. (2021)	✓	dialogue	✗	✓
Qi et al. (2022)	✓	writings	✗	✓
MacLaughlin et al. (2021)	✓	writings	✓	✗
Our dataset	✓	writings	✓	✓

Table 2: Dataset comparison for other related tasks with ours.

overall retriever-reader architecture RA-MTR is illustrated in the left part of Figure 3.

5.1 Retrieval augmented multi-task reader (RA-MTR)

5.1.1 Retriever

In the preprocessing stage, we divide each book into fixed-length paragraphs, and our quote prediction module attempts to identify the relevant span of a quote from the positive paragraph. However, in real-world scenarios, we are not provided with the positive paragraph, instead, we have a large set of paragraphs from which we need to retrieve the positive paragraph. Retrieving the positive paragraph from a set of non-relevant paragraphs is a challenging task. Inspired from RAG (Lewis et al., 2020) architecture, we employ a vector-store based retriever to initially retrieve top-k[8] ($k = 20$) paragraph based on the given context. We utilize *langchain* API[9], to split the source document into several chunks (we choose chunk-size of 1200[10] characters and chunk-window of 100), followed by encoding of each chunk using *sentence-transformers*, and finally store the embeddings into a vector-store for efficient searching. We use ChromaDB[11] for storing the embeddings of the chunks. In parallel, the query context is also embedded using *sentence-transformers*. To extract the relevant inspirational quote from the source document we perform a similarity search by comparing the query context embedding and the embeddings in the vector-store to retrieve top-k chunks. The retrieved chunks are then passed to the more powerful sequence-to-sequence re-ranking module for further processing.

Fine-tuning paragraph re-ranking module: After retrieving initial sets of candidate paragraphs, many past literature leveraged deep neural network based paragraph re-ranking modules to get a final ranked list of paragraphs. Works such as (Dai and Callan (2019); Yilmaz et al. (2019); Nogueira et al. (2019)) exploited BERT for paragraph/document retrieval tasks. Nogueira et al. (2020) used a T5-ased encoder-decoder architecture for document ranking. We apply a more sophisticated decoder-only transformer based model *Llama-3*[12] to re-rank the paragraph. Similar to Nogueira et al. (2020) we formulate the problem as a binary classification task, and the input prompt is:

```
Context: {c}
Document: {d}
Is the document relevant to the context? Answer yes/no:
```

where c and d are the context and paragraph texts, respectively. The model is fine-tuned to produce the words yes or no depending on whether the document is relevant or not to the query. That is, yes and no are the 'target words' (i.e., ground truth predictions in the sequence-to-sequence transformation). To generate training and test examples for the models, we iterate over each context and create (context, source paragraph, label) example triples for each paragraph in the corresponding source document. The label is yes if the author actually quoted from the paragraph (positive triple) and no (negative triple) otherwise. At inference time, to compute probabilities for each query–document pair (in a re-ranking setting), we retrieve the unprocessed next-token probabilities for the tokens yes and no. From these, we calculate the $yes - score$ as follows.

$$yes - score_{(c,d_i)} = \frac{p(yes|P)}{p(yes|P) + p(no|P)} \quad (1)$$

where, c is the context, d_i is the i^{th} document and P is the prompt. Similarly, as baseline, we also fine-

[8]Increasing k did not change the performance too much

[9]https://python.langchain.com/docs/modules/data_connection/

[10]As we find the maximum length of context + paragraph is 1005

[11]https://pypi.org/project/chromadb/

[12]We apply the chat model from huggingface meta-llama/Meta-Llama-3-8B-Instruct

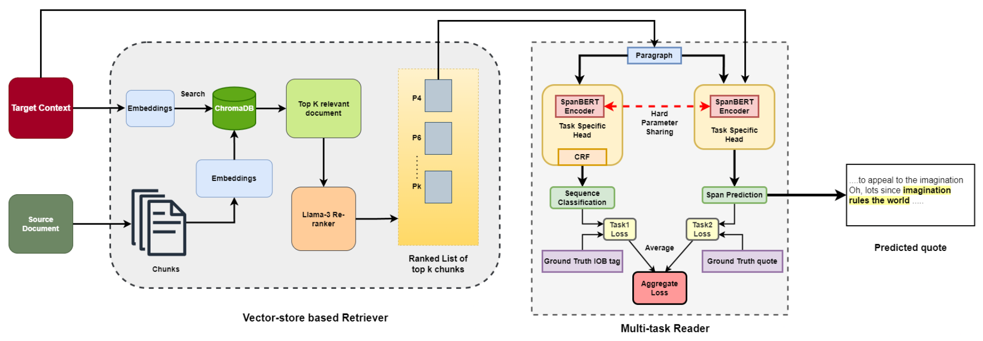


Figure 3: The RA-MTR architecture.

tune encoder-decoder based models (T5, FLAN-T5) for the re-ranking task using similar approach. *Sampling hard negatives*: Out of all the negative triples obtained we select n hard samples for training to make the model more robust. We explore two different hard negative sampling methods - a) select the paragraphs that are closest next to the positive paragraph, and b) select the top ranked paragraphs (other than the positive one) returned by BM25 retriever model. However, we observe that both choices produce similar results (see results section for details).

5.1.2 Multi-task reader for quote extraction

Motivation for multi-task training: Unlike normal spans of text, quotes have certain inherent special properties or some figurative language that make them unique (Bendersky and Smith, 2012). We believe that identifying such special occurrences of phrases is essential for quote prediction from paragraphs. We cast this as a sequence classification, i.e., marking only the portions of a text that can be recounted as quotable. We attempt to optimize two tasks in parallel - quotable sequence identification (using SpanBERT-CRF) and context awareness (using SpanBERT). In a paragraph, there can be multiple spans of text which will be relevant to the context. However, not every relevant span is quotable. The span prediction module is essentially a variant of a question-answering module, which might not be good enough to identify quotability of the answer. Also, many of the quotes are only subparts of a sentence (e.g., “He travels fastest who travels alone,...”) while few of the quotes consist of more than one sentence (e.g., “In this world there are only two tragedies. One is not getting what one wants, and the other is getting it.”). To mitigate this gap, we use a specific sequence identification module (SpanBERT-CRF) to find quotable sequences.

Span prediction from paragraph: We train the span prediction model using context-quote-paragraph triple as the training example. Similar to (MacLaughlin et al., 2021), we utilize the span-level BERT architecture, which receives the packed sequence of the context and paragraph as input. By utilizing the final hidden vector $T_i \in \mathbb{R}^h$ as the representation for each wordpiece in a given paragraph, we follow the standard approach of casting span prediction as two classification tasks, i.e., separately predicting the start and end of the span (Devlin et al., 2019). To this purpose, we introduce a start vector, $S \in \mathbb{R}^h$, and an end vector, $E \in \mathbb{R}^h$. The probability of a word w being the start of the quoted span is the dot product $S \cdot T_w$ followed by a softmax over all wordpieces in the example. We follow the same approach for calculating the probability of word w being the end of the span using E. The loss is calculated as the average NLL (Negative log-likelihoods) of the correct start and end positions, i.e., the tokens in the paragraph the author actually quoted. Following (Devlin et al., 2019), at prediction time, the score of the span from position i to j is $S \cdot T_i + E \cdot T_j$. We consider a span valid if $j > i$ and i and j occur in the paragraph portion of the input. We retain a mapping from wordpieces to original tokens for prediction.

Quotability identification as sequence classification: We take inspiration from (Portelli et al., 2021), which used sequence labeling for the adverse drug events (ADE) detection from a given text. Along similar lines, we employ SpanBERT neural model combined with Conditional Random Field (CRF) to identify quotable phrases or words. Each example from the dataset is accompanied by a paragraph, the start and end character positions of the quote in that paragraph. Using this information, we first convert this into the commonly used IOB (Inside, Outside, Beginning) schema using *Spacy*. Consider the example in Figure 1, every word except the bold portion (i.e., the quote) should be marked as ‘O’. The word ‘hopeless’ in the quote should be labeled as ‘B’ and the rest of the quote should be labeled as ’I’. The BIO tagging is illustrated in Figure 4. Since BERT-based models

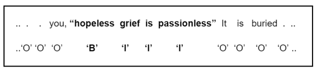

.. . . you, **“hopeless grief is passionless”** It is buried . ..
..‘O’ ‘O’ ‘O’ **‘B’** **‘I’** **‘I’** **‘I’** ‘O’ ‘O’ ‘O’ ‘O’ ..

Figure 4: Example of BIO tagging.

generally employ wordpiece tokenizers to tackle the out-of-vocabulary words, which actually break such words into multiple subwords, we require to decide on a consistent IOB schema for the subwords. We set a rule for the sub-labels which are consistent with the IOB schema: words labeled as B generate a series of subwords labeled as [B, I, . . . , I], while words labeled as I (or O) generate a series of identical I (or O) sub-labels. For example, if the word ‘resisted’ has the label B, then its corresponding wordpieces - [‘Resist’, ‘##ed’] would get labeled as [B, I].

Multi-task training: To take advantage of both the span prediction model and the quotable phrase identification model, we adopt a multi-task based framework where we have two independent models and they share the same transformer encoder. The span prediction model tries to match the start and end token of the quote in the paragraph, whereas the quotable phrase identification model tries to predict the ‘B’, ‘I’, and ‘O’ labels for each token. During the backpropagation, we average the losses from the two models. The right part in the Figure 3 demonstrates the architecture of the multi-task framework.

6 Experiments

In this section, we discuss the experiments that we conduct and the details of the experimental setups.

Fine-tuning paragraph re-ranking: We pass the packed input of context and paragraphs to the retriever model. Out of **4889** data points in the QuoteR dataset, and we select 80% for training, 10% each for dev and test. We choose to fine-tune the *Llama-3-8b-instruct* model for the paragraph ranking task. For model implementation details and hyper-parameters see Appendix F.

Fine-tuning reader: We fine-tune the reader by randomly selecting 80% QuoteR data for training, 10% each for dev and test (see Appendix F for implementation details). To test the generalizability of the model in a few-shot setting, we consider random training samples $\in \{4, 8, 16, 32, 64\}$ from the other two datasets (i.e., Gandhi and Quotus) for further fine-tuning with a slightly lower learning rate ($1e^{-5}$), and test on the remaining data samples for the respective datasets.

Metrics: Since the setup for our span prediction task is identical to QA, we evaluate the span-level models using the two popular QA metrics – (i) exact match (EM), and (ii) macro-averaged bag-of-words (BoW) F1. EM measures if the predicted span exactly matches the positive quote, and BoW-F1 measures their average word overlap.

Baselines: Both the retriever and the reader can have many variants which serve as ideal baselines. In the retriever part we use vanilla BM25 as a first baseline. Apart from the simple BM25 retriever, we employ encoder-based (BERT), encoder-decoder based (T5, FLAN-T5) document re-ranking to improve paragraph selection.

For the *reader* part, as primitive baselines, we consider using the first and last sentences of each paragraph as potential quotes. To further explore, we also fine-tune the BERT and the SpanBERT pretrained models on the BERT question answering architecture. We keep the same hyperparameter settings as the multi-task framework. Additionally, we also observe the ability of different medium sized open-source LLMs such as FLAN-T5-large[13], FLAN-T5-XL[14], Bloomz-3b[15], Falcon-7b[16], Llama-3-8b[17] models for zero-shot context-aware quote extraction task. For the detailed methodology, refer to Appendix E.

7 Results

Performance of RA-MTR: To examine the efficacy of our entire pipeline, we conduct an end-to-end prediction from our approach. In the *retriever-reader* based (baseline) approach, we first provide the context and the list of paragraphs segmented from a particular book to the paragraph retrieval module. We initially get a list of 20 top-ranked paragraphs relevant to the context from the RAG model and then re-rank these using the Llama-3 model. We take the top three paragraphs further and sequentially pass them with respect to the con-

[13]https://huggingface.co/google/flan-t5-large

[14]https://huggingface.co/google/flan-t5-xl

[15]https://huggingface.co/bigscience/bloomz-3b

[16]https://huggingface.co/tiiuae/falcon-7b

[17]https://huggingface.co/meta-llama/Meta-Llama-3-8B-Instruct

text to our multi-task quote extraction module. This span prediction module within the multi-task framework predicts the top three quotable spans, each from one corresponding paragraph. We measure the BoW F1-score for these three predictions with respect to the ground truth quote and report the scores for 1) top-1 prediction - score when we compare the ground truth with the predicted span from the top 1 out of the three ranked paragraphs, and 2) top-3 predictions - best score when we compare the ground truth with all the three predictions. Table 3 demonstrates the result for the end-to-end quote prediction RA-MTR. We compare the performance of our pipeline with two commonly used baselines for open-domain question answering tasks – DrQA (Chen et al., 2017) and ParagraphRanker (Lee et al., 2018). We also compare RA-MTR against (MacLaughlin et al., 2021), which focuses on context-based quote extraction. Lastly, we compare our model with present day LLM variants. RA-MTR by far outperforms all the baselines.

Multi-task reader performance: We show the results for span prediction using various methods in Table 4 for the QuoteR dataset. The results in the first two rows are from two very primitive baselines. Scores in the next two rows are from only the span selection models, which (MacLaughlin et al., 2021) has considered. We can clearly see that our multi-task based approach outperforms the other methods. The improvements are significant with $p < 0.05$ as per the Mann-Whitney U test (Mann and Whitney, 1947) and experiment with our three datasets. In Table 5, we demonstrate the few-shot performances on the Gandhi and the Quotus data for the quote prediction task. We can observe that, in the few-shot settings, the multi-task framework performs much better than the simple span prediction models that are normally used for the QA tasks. In fact, with only 8 data samples from the target domain, our model beats BERT and SpanBERT by **14%** and **11%** for the Gandhi data, and by **7%** and **4%** for the Quotus data respectively. We can infer from these results that the addition of the quotable phrase identification task actually helps the model learn the linguistic properties of the quotes much better than the simple span prediction model. Further, the multi-task framework generalizes particularly well in the cross-domain setting even with the training and test paragraphs coming from different genres.

Performance of the sequence classifier: We analyze the output generated by the sequence classifier head from our multi-task framework. Note that this was an auxiliary task to improve the main task of span prediction. The sequence classifier head typically predicts ‘B’, ‘I’, or ‘O’ tags for each token, and the prediction is independent of the context. We apply the model to instances in the QuoteR test dataset and extract the predicted labels from the sequence classification head. We find that the model correctly predicts the BIO labels for 48.1% of the instances. In 20.7% of the cases, the model predicts multiple BIO labels within a single paragraph, indicating that one paragraph can contain multiple instances of quotable phrases.

Context (in)dependence: We conduct an ablation experiment to observe the impact of context for the quote prediction in the multi-task setting. We remove the whole context from the inputs in the test set for the quote prediction models while experimenting with the QuoteR dataset[18]. Table 6 clearly shows that the baseline models’ performances are drastically reduced, whereas our multi-task framework outperforms the two baselines. As the sequence classification task is independent of the context we observe that the multi-task framework performs better in the absence of context while the two other models that are highly context-sensitive. We can infer that the linguistic boundary identification for the quotes in terms of the BIO markers enhances the performance and makes it robust to the absence of context. This is one of the prime strengths of the multi-task framework.

8 Analysis of the multi-task reader output

Analysis of top predicted quotes: Since there may be multiple quotes in a given paragraph for a given context, we also look at the top five predicted spans from our multi-task framework for each of the paragraphs in the test set. We manually annotate the relevance of the predicted spans for the top five predictions. We had two annotators, and each of them was provided with a set of context and the top five predicted spans. They were required to mark 1 if the predicted span is semantically coherent with the context, and 0 otherwise. In the case of ambiguity, a third annotator was involved to adjudicate. We obtain an inter-annotator agreement of Cohen’s $\kappa = 0.64$. We take the final relevancy (i.e., 0 or 1) based on majority vote. We achieve a high MAP@5 score of 0.78, indicating that our

[18]The results from the other datasets show similar trends and hence are not shown.

Method	Dataset					
	QuoteR (test)		Gandhi		Quotus	
	Top-1 F1	Top-3 F1	Top-1 F1	Top-3 F1	Top-1 F1	Top-3 F1
DrQA (Chen et al., 2017)	7.19	8.22	6.20	8.38	4.26	5.43
ParagraphRanker (Lee et al., 2018)	16.58	21.45	12.17	14.35	11.31	15.11
BM25 + (MacLaughlin et al., 2021) (Positive only settings)	31.78	37.37	23.70	26.60	32.58	37.68
BM25 + BERT-base + MTR*	37.20	46.28	25.17	27.30	36.15	41.12
BM25 + T5-base + MTR*	34.17	45.21	21.31	23.45	37.13	39.25
BM25 + T5-large + MTR*	39.07	48.29	28.13	30.67	39.29	41.11
BM25 + FLAN-T5-large + MTR*	43.12	51.43	34.46	42.30	42.26	48.21
Vector-store based retriever + LLM reader (Llama-3-8b-instruct)	14.81	18.76	17.53	23.28	31.38	39.77
RA-MTR (ours)	**45.74**	**57.25**	**38.71**	**49.38**	**43.40**	**53.45**

Table 3: The result for the quote extraction using the different baseline models and our RA-MTR approach. For a fair comparison, we took the results from the positive-only settings of (MacLaughlin et al., 2021). Note that all the fine-tuned models are only trained on the QuoteR training data. *MTR: Our fine-tuned multi-task reader.

Method	EM	BoW-F1
First sentence	0.55	6.31
Last sentence	1.08	5.88
BERT-base	69.1±0.5	76.2±0.9
BERT-large	71±0.3	77.9±0.3
SpanBERT-base	71.7±0.6	77.7±0.4
SpanBERT-large	72.3±0.6	79.2±0.4
Multi-task using SpanBERT-base (Ours)	73±0.8	78.2 ±0.3
Multi-task using SpanBERT-large (Ours)	**77±0.4**	**86.1 ±0.2**

Table 4: Reader performance on the QuoteR dataset. We provide the positive paragraph to predict the quote span.

Test data	# training samples	Methods		
		BERT	SpanBERT	Multi-task (Ours)
Gandhi	8	27.71	30.30	41.32
	16	32.60	32.65	50.29
	32	38.12	36.85	62.91
	64	43.54	44.65	72.08
Quotus	8	33.97	36.82	40.58
	16	37.90	41.84	49.33
	32	40.56	44.80	55.08
	64	47.86	51.27	59.12

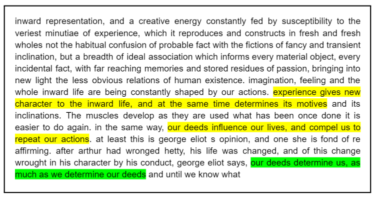
Table 5: Few-shot inference performance on the 1) Gandhi and 2) Quotus datasets. We have used the BoW F1-score as the metric for comparison here.

Method	EM	BoW-F1
BERT span prediction	19.20±0.30	30.90±0.80
SpanBERT span prediction	18.30±0.50	29.70±0.40
Multi-task (Ours)	22.00±0.80	38.20±0.70

Table 6: Results for the quote extraction in absence of the context (for QuoteR dataset).

multi-task framework retrieved $\sim$ 3.9 (on average) meaningful recommendations among the top five recommendations.

Error in the sequence tagger: In order to understand the reasons behind the incorrect predictions made by the model, we review some instances where the model failed to predict the correct BIO labels. A specific example is depicted in Figure 5, where the true quote is highlighted in green, while the predicted quotes are highlighted in yellow. Despite the fact that the true quote and the predicted quotes come from different portions of the paragraph, they all possess high quotability as per human experts. We observe many such cases of (pseudo) errors that manifest due to the absence of valid additional ground truth quotes.



inward representation, and a creative energy constantly fed by susceptibility to the veriest minutiae of experience, which it reproduces and constructs in fresh and fresh wholes not the habitual confusion of probable fact with the fictions of fancy and transient inclination, but a breadth of ideal association which informs every material object, every incidental fact, with far reaching memories and stored residues of passion, bringing into new light the less obvious relations of human existence. imagination, feeling and the whole inward life are being constantly shaped by our actions. experience gives new character to the inward life, and at the same time determines its motives and its inclinations. The muscles develop as they are used what has been once done it is easier to do again. in the same way, our deeds influence our lives, and compel us to repeat our actions. at least this is george eliot s opinion, and one she is fond of re affirming. after arthur had wronged hetty, his life was changed, and of this change wrought in his character by his conduct, george eliot says, our deeds determine us, as much as we determine our deeds and until we know what

Figure 5: Example of an instance where the sequence classifier wrongly predicts the BIO labels. The true quote is highlighted in green, while the predicted quotes are highlighted in yellow.

Error in the multi-task reader: In this segment, we attempt to analyze the quotes predicted by our multi-task framework. We examine the predicted quotes, which do not entirely match with the ground truth quotes. We observe that in most of the cases (72%), the model predicts a sub-phrase of the original quote. For instance, while the actual quote is ‘Our Father, which art in heaven, hallowed be thy name’, the corresponding predicted quote is ‘which art in heaven, Hallowed be thy Name’. In a few cases, the model over-predicts, i.e., predicts a span containing the true quote and some phrases surrounding it. For example, the actual quote ‘Money begets money’, is predicted as ‘Money begets money and its offspring’.

9 Conclusion

In this work, we proposed a method to recommend quotes from large texts given a context. We employed a novel multi-task framework for quote prediction, which can in parallel predict the span of text and identify the quotable phrases. We constructed three datasets of different genres and experimented on them. We believe that our methodology and datasets will be beneficial for future research.

Limitations

In this section we will discuss the limitations of our study. While it is evident that the quotes are available in different regional languages, all of our experiments are conducted for the English version of the datasets. Few of the pre-processing steps might not be suitable for the languages with different morphosyntactic structures. Further the base models will also need to be changed.

Ethics Statement

We used three datasets for our experiments. The QuoteR dataset was released publicly by the authors of (Qi et al., 2022). Besides, we extracted all the paragraphs from open corpora, including free public domain e-books. The quotes of Gandhi were collected from the free quote repository and the context were extracted from the publicly available portal. Both the quotes and the contexts for the Quotus data were collected from the open corpora. The annotators voluntarily annotated the predictions for our analysis, and we did not retain any of their private information.

References

Sayantan Adak, Atharva Vyas, Animesh Mukherjee, Heer Ambavi, Pritam Kadasi, Mayank Singh, and Shivam Patel. 2020. Gandhipedia: A one-stop ai-enabled portal for browsing gandhian literature, life-events and his social network. In *Proceedings of the ACM/IEEE Joint Conference on Digital Libraries in 2020*, JCDL '20, page 539–540, New York, NY, USA. Association for Computing Machinery.

Michael Bendersky and David Smith. 2012. A dictionary of wisdom and wit: Learning to extract quotable phrases. In *Proceedings of the NAACL-HLT 2012 Workshop on Computational Linguistics for Literature*, pages 69–77, Montréal, Canada. Association for Computational Linguistics.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading Wikipedia to answer open-domain questions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1870–1879, Vancouver, Canada. Association for Computational Linguistics.

Zhuyun Dai and Jamie Callan. 2019. Deeper text understanding for ir with contextual neural language modeling. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR'19, page 985–988, New York, NY, USA. Association for Computing Machinery.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. 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. Association for Computational Linguistics.

Aaron Gokaslan and Vanya Cohen. 2019. Openwebtext corpus. `http://Skylion007.github.io/OpenWebTextCorpus`.

Miguel Grinberg. 2018. *Flask web development: developing web applications with python*. " O'Reilly Media, Inc.".

Fajri Koto, Sakriani Sakti, Graham Neubig, Tomoki Toda, Mirna Adriani, and Satoshi Nakamura. 2014. Memorable spoken quote corpora of ted public speaking. In *2014 17th Oriental Chapter of the International Committee for the Co-ordination and Standardization of Speech Databases and Assessment Techniques (COCOSDA)*, pages 1–4.

Hanbit Lee, Yeonchan Ahn, Haejun Lee, Seungdo Ha, and Sang-goo Lee. 2016. Quote recommendation in dialogue using deep neural network. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '16, page 957–960, New York, NY, USA. Association for Computing Machinery.

Jinhyuk Lee, Seongjun Yun, Hyunjae Kim, Miyoung Ko, and Jaewoo Kang. 2018. Ranking paragraphs for improving answer recall in open-domain question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 565–569, Brussels, Belgium. Association for Computational Linguistics.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474. Curran Associates, Inc.

Yuanchao Liu, Bo Pang, and Bingquan Liu. 2019. Neural-based Chinese idiom recommendation for enhancing elegance in essay writing. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5522–5526, Florence, Italy. Association for Computational Linguistics.

Ansel MacLaughlin, Tao Chen, Burcu Karagol Ayan, and Dan Roth. 2021. Context-based quotation recommendation. *Proceedings of the International AAAI Conference on Web and Social Media*, 15(1):397–408.

Ansel MacLaughlin and David Smith. 2021. Content-based models of quotation. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 2296–2314, Online. Association for Computational Linguistics.

Henry B Mann and Donald R Whitney. 1947. On a test of whether one of two random variables is stochastically larger than the other. *The annals of mathematical statistics*, pages 50–60.

Vlad Niculae, Caroline Suen, Justine Zhang, Cristian Danescu-Niculescu-Mizil, and Jure Leskovec. 2015. Quotus: The structure of political media coverage as revealed by quoting patterns. In *Proceedings of the 24th International Conference on World Wide Web*, pages 798–808.

Rodrigo Nogueira, Zhiying Jiang, Ronak Pradeep, and Jimmy Lin. 2020. Document ranking with a pre-trained sequence-to-sequence model. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 708–718, Online. Association for Computational Linguistics.

Rodrigo Nogueira, Wei Yang, Kyunghyun Cho, and Jimmy Lin. 2019. Multi-stage document ranking with bert. *arXiv preprint arXiv:1910.14424*.

Beatrice Portelli, Daniele Passabì, Edoardo Lenzi, Giuseppe Serra, Enrico Santus, and Emmanuele Chersoni. 2021. Improving adverse drug event extraction with spanbert on different text typologies. In *International Workshop on Health Intelligence*, pages 87–99. Springer.

Fanchao Qi, Yanhui Yang, Jing Yi, Zhili Cheng, Zhiyuan Liu, and Maosong Sun. 2022. Quoter: A benchmark of quote recommendation for writing. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 336–348.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(1).

Jiwei Tan, Xiaojun Wan, Hui Liu, and Jianguo Xiao. 2018. Quoterec: Toward quote recommendation for writing. *ACM Transactions on Information Systems (TOIS)*, 36(3):1–36.

Jiwei Tan, Xiaojun Wan, and Jianguo Xiao. 2015. Learning to recommend quotes for writing. *Proceedings of the AAAI Conference on Artificial Intelligence*, 29(1).

Jiwei Tan, Xiaojun Wan, and Jianguo Xiao. 2016. A neural network approach to quote recommendation in writings. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, CIKM '16, page 65–74, New York, NY, USA. Association for Computing Machinery.

Nikos Voskarides, Edgar Meij, Sabrina Sauer, and Maarten de Rijke. 2021. News article retrieval in context for event-centric narrative creation. In *Proceedings of the 2021 ACM SIGIR International Conference on Theory of Information Retrieval*, pages 103–112.

Lingzhi Wang, Xingshan Zeng, and Kam-Fai Wong. 2021. Quotation recommendation and interpretation based on transformation from queries to quotations. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 754–758.

Zhiguo Wang, Patrick Ng, Xiaofei Ma, Ramesh Nallapati, and Bing Xiang. 2019. Multi-passage bert: A globally normalized bert model for open-domain question answering. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5878–5882.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Zeynep Akkalyoncu Yilmaz, Shengjin Wang, Wei Yang, Haotian Zhang, and Jimmy Lin. 2019. Applying bert to document retrieval with birch. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP): System Demonstrations*, pages 19–24.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE international conference on computer vision*, pages 19–27.

A Analysis of retriever

In this section, we attempt to analyse how well our vector-store based retriever performed. As we cannot directly compare the retrieved chunks with the positive paragraph in our dataset (due to variable word length), we measure the average *Jaccard* similarity between the top predicted chunk with the positive paragraph in our dataset for a specific context. We present the results in Figure 6. We observe that, using Llama-3 based re-ranking, the similarity significantly improved for all the three datasets.

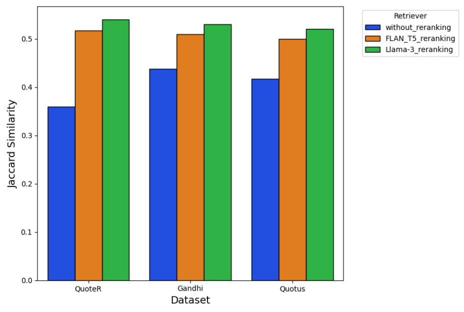


Figure 6: Average *Jaccard* similarity between top predicted chunk and positive paragraph for a specific context.

B Results using different LLMs as reader

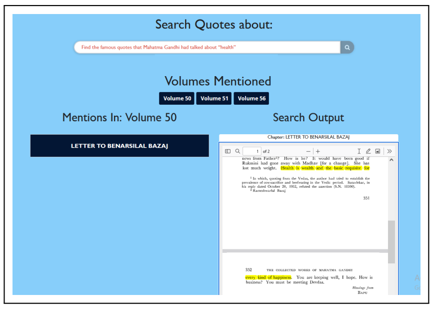
Extending the Table 3, we demonstrates the results while using different other LLMs.

C Deployment status

We have deployed the RA-MTR framework in a flask (Grinberg, 2018) based web application (link will be made public upon acceptance). We plan to integrate this system with the publicly available and fully searchable historical encyclopedia (e.g., Gandhipedia[19]). We present an example page of our demo system in Figure 7. The figure shows the result when a user searches for the query "Find the famous quotes that Mahatma Gandhi had made about *health*". The system extracts the most relevant quotes from the entire 100 volumes of the Collected Works of Mahatma Gandhi and highlights them in yellow.

D Data preprocessing details

QuoteR data : The *Project Gutenberg corpus* comprises more than 73000 e-books in textual form. We assign each of these books a unique bookID and divide each book into fixed-length (i.e., 200 word length) paragraphs, and assign each such paragraph a unique paragraphID. The distribution of the number of paragraphs per book and the number of tokens per quote is presented in Figure 8 and 9. We construct a TF-IDF weighted word-doc sparse matrix (Chen et al., 2017) from all the documents, index, and store this content in the `sqlite db`. For each of the quotes present in the QuoteR dataset we recursively search for the appearance of the quote in each of these books. Once a search gets a hit, we link the bookID with that particular quote (to be used for training the paragraph retrieval model). Since the authors in (Qi et al., 2022) stored the context from different sources and the correct mapping to the books is not present, we consider the 40 words preceding and following it as its left and right contexts, respectively. Similar to (Qi et al., 2022) the concatenation of the left and right contexts forms a complete context. We then store the context, quote, and positive paragraph (to be used for training the quote prediction model). Out of the 6108 unique quotes, we are able to find the occurrences of 1708 quotes from the *Project Gutenberg* and we finally construct 4889 quote-context-paragraph (one quote may contain multiple contexts) triples as examples for training and evaluating. The algorithm for generating the quote-context-paragraph triples is presented in Algorithm 1.


Search Quotes about:

Find the famous quotes that Mahatma Gandhi had talked about "health"

Volumes Mentioned

Volume 50 Volume 51 Volume 56

Mentions In: Volume 50

LETTER TO BENARSILAL BAZAJ

Search Output

Chapter: LETTER TO BENARSILAL BAZAJ

1 of 2

news from Father[2]? How is he? It would have been good if Rukmini had gone away with Madhav [for a change]. She has lost much weight. Health is wealth and the basic requisite for

[1] In which, quoting from the Vedas, the author had tried to establish the prevalence of cow-sacrifice and beef-eating in the Vedic period. Satavlekar, in his reply dated October 20, 1932, refuted the assertion (S.N. 18590).
[2] Rameshwarlal Bazaj

351

352 THE COLLECTED WORKS OF MAHATMA GANDHI

every kind of happiness. You are keeping well, I hope. How is business? You must be meeting Devdas.

Blessings from
BAPU

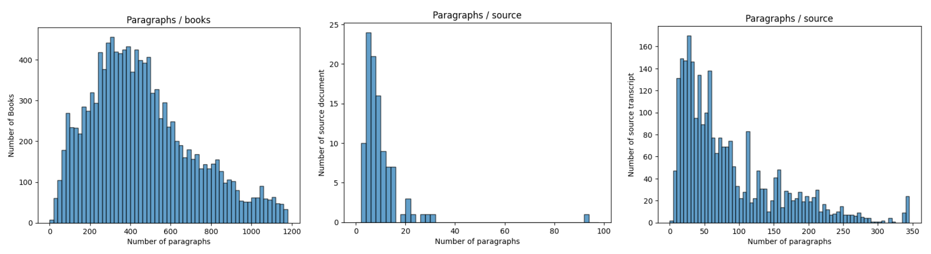
Figure 7: Example of a real time quote extraction from the Collected Works of Mahatma Gandhi. The output quote is highlighted in yellow in the pdf.

Gandhi data : Similarly, for the Gandhi quotes, we search for the quotes in the CWMG and find their appearance in a particular chapter of a book in the CWMG. We utilize the *Gandhipedia* (Adak et al., 2020) engine, which uses an elasticsearch based search engine to locate the quotes. We consider the 40 preceding and following words from

[19] https://gandhipedia150.in/en/

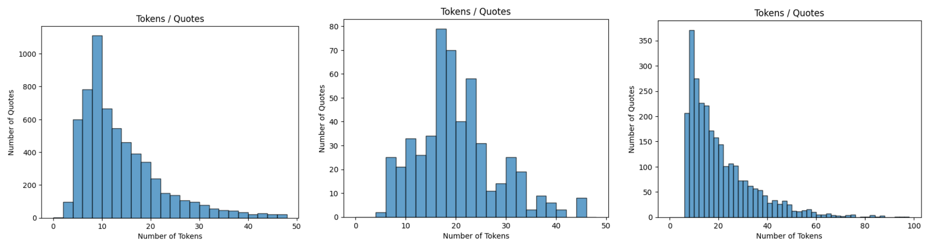
Method	Dataset					
	QuoteR (test)		Gandhi		Quotus	
Vector-store based retriever + LLM reader (FLAN-T5-large)	14.5	19.23	18.2	22.5	25.27	31.2
Vector-store based retriever + LLM reader (FLAN-T5-xl)	13.32	19.4	16.54	24.33	35.0	38.25
Vector-store based retriever + LLM reader (bloomz-3b)	10.33	12.12	9.19	13.48	16.43	21.31
Vector-store based retriever + LLM reader (Falcon-7b)	10.08	13.34	17.05	22.35	29.73	36.73
Vector-store based retriever + LLM reader (Llama-3-8b-instruct)	14.81	18.76	17.53	23.28	31.38	39.77

Table 7: The result for the quote extraction using the different LLMs as reader



(a) QuoteR (b) Gandhi (c) Quotus

Figure 8: # of paragraphs per source documents.



(a) QuoteR (b) Gandhi (c) Quotus

Figure 9: Distributions over source document, paragraphs, and quote lengths.

each quote in the particular chapter as its context. In addition, we find that out of all the Gandhi quotes, three quotes are already present in the QuoteR set. We, therefore, remove them from the Gandhi data. Finally, we obtain 737 quote-context-paragraph triples.

Generating target context : Unlike the Quotus data, we do not have explicit target documents (i.e., where the quote needs to be recommended from source) for the QuoteR and Gandhi dataset. We synthesize the target context by paraphrasing the original context in the corresponding dataset. This is performed to reduce the overlapping words of the target context and the source document and to effectively evaluate the robustness of the methodology. We use ChatGPT[20] API to paraphrase the context. The examples of such paraphrased context are provided in Appendix G. The prompt used for paraphrasing:

```
As a paraphrasing expert can you rephrase
the following input text? Ensure the
rephrased text incorporates a differ-
ent range of vocabulary compared to the
original text.
Input text: {<Input text>}
Rephrased text:
```

To analyse the hardness of the generated target contexts, we measure the word overlap between the original context and the rephrased context. We observe that the average word overlap ratio between the original and the rephrased contexts are - 0.19 and 0.18 for QuoteR and Gandhi data respectively. This indicates that the rephrased target context has significantly different words thus making the task of the paragraph retriever harder.

Quotus data : For the Quotus dataset, we utilize the Quoting POTUS website[21] to collect a set of examples for our experiments. They release the transcripts and the collection of aligned quotes, containing the text of the quote in the news article, its aligned position within the source transcript, and the corresponding news article metadata (title,

[20]https://openai.com/blog/chatgpt

[21]http://snap.stanford.edu/quotus/vis/

url, timestamp). We crawl the provided news article URLs and extract the body content of each news article using BeautifulSoup[22]. We are able to extract 10,114 news articles in this way (some of the links were not working and could not be crawled). To locate the quotes within the news articles, we utilize regular expressions and identify the appearance of 2,698 quotes. We then consider the 40 preceding and following words from each quote in the news article as its context. In the released dataset, the source transcript is already divided into several paragraphs, and the alignment of the quotes to the positive paragraph is also provided. As a result, we did not need to explicitly create the quote-paragraph alignment. This yields a total of 2,698 quote-context-paragraph triples, which we use for our experiments.

The Algorithm 1 shows the step-by-step procedure to prepare the dataset for our experiments. The auxiliary functions (i.e., Algorithms 3, 4 and 2) used in the algorithms are depicted in the subsequent algorithms.

E Baseline methods

Baselines: Both the retriever and the reader can have many variants which serve as ideal baselines. In the retriever part we use vanilla BM25 as a first baseline. Apart from the simple BM25 retriever, we employ BERT and T5 based re-ranking to improve paragraph selection. For input to BERT we tokenize the contexts and source document paragraphs into wordpieces (Wu et al., 2016) and cap them at predetermined lengths chosen as hyperparameters. BERT uses a special token [SEP] to separate paragraph from the context. So the final wordpiece input to the BERT is:

$$[CLS] \quad context \quad [SEP] \quad paragraph \quad [SEP]$$

Following (Wang et al., 2019), we fine-tune BERT-base using the pairwise loss. Thus, a single training example for paragraph BERT consists of $n + 1$ instances, i.e., one positive instance plus n negative instances. Each of the $n+1$ packed input sequences are fed to BERT independently. We use the final hidden vector $\mathbf{C} \in \mathbb{R}^h$ corresponding to the first input token [CLS] as the representation for each of the $n + 1$ sequences, where h is the size of the final hidden layer. In addition, we also fine-tune encoder-decoder based (T5, FLAN-T5) and decoder-only (Llama-3) re-ranking models in the same way as discussed in section 5.1.1.

For the reader part, as primitive baselines, we consider using the first and last sentences of each paragraph as potential quotes. To further explore, we also fine-tune the BERT and the SpanBERT pre-trained models on the BERT question answering architecture. We keep the same hyperparameter settings as the multi-task framework. Again, we fine-tune on 80% of the QuoteR data, and use 10% for validation before testing on the remaining 10%. In addition, we conduct similar few-shot experiments with the Gandhi and the Quotus dataset.

LLM based baselines: With the advancement of large language models (LLMs) such as T5 (Raffel et al., 2020), GPT-3 (Brown et al., 2020) it is important to observe their ability to perform the task of quote extraction. These models have proven to be highly valuable for contextual learning when provided with specific prompts in zero-shot scenarios. We replace the multi-task reader with different medium sized open-source LLMs such as FLAN-T5-large[23], FLAN-T5-XL[24], Bloomz-3b[25], Falcon-7b[26], Llama-3-8b[27] models to predict the most relevant quote given the paragraph and context. We use the below prompt:

```
You are an AI assistant in recommending a suitable 'quote' based on the context and your task is to extract a relevant quote from the given pargraph based on the context. Note that, the context and the paragraph may contain grammatical errors. DO NOT use any external information.

Context: "{context}"

Paragraph: "{paragraph}"

Just extract the relevant quote without any other sentence:
```

F Model implementation details

Retriever : For retriever we use *lagchain API*[28], employ recursive_text_splitter[29] for splitting the

[22]https://www.crummy.com/software/BeautifulSoup/bs4/doc/

[23]`https://huggingface.co/google/flan-t5-large`

[24]`https://huggingface.co/google/flan-t5-xl`

[25]`https://huggingface.co/bigscience/bloomz-3b`

[26]`https://huggingface.co/tiiuae/falcon-7b`

[27]`https://huggingface.co/meta-llama/Meta-Llama-3-8B-Instruct`

[28]`https://python.langchain.com/docs/modules/data_connection/`

[29]`https://python.langchain.com/docs/modules/data_connection/document_transformers/recursive_`

Algorithm 1 Paragraph retrieval data generation

```
Require: list_of_quotes: list of selected quotes; corpus_directory: directory of the corpus (ex. Gutenberg)
 1: $quoteid\_book\_mapping \leftarrow$
    $CREATE\_QUOTE\_TO\_BOOK\_MAPPING(list\_of\_quotes, corpus\_directory)$
 2: $ctxid \leftarrow 0$                                        ▷ Initialize Context Id
 3: $ctxid\_to\_text \leftarrow \{\}$                          ▷ Initialize Context Id to Context text mapping
 4: $quoteid\_to\_ctxid \leftarrow \{\}$                       ▷ Initialize Quote Id to Context Id mapping
 5: for all ($quoteid$, list_of_book_paths) in $quoteid\_book\_mapping$ do
 6:     $dataset \leftarrow []$                                ▷ Dataset to be used for training and testing paragraph retrieval
 7:     $quoteid\_to\_ctxid[quoteid] \leftarrow []$
 8:     for all $book\_path$ in $list\_of\_book\_paths$ do
 9:         $paragraphs \leftarrow$ SEGMENT_BOOK$(book, paragraph_lenght = 200)$   ▷ Segment the book contents into several paragraphs
10:         save$(docid\_to\_text)$
11:         for all $paragraph$ do
12:             if $quote$ in $paragraph$ then
13:                 $ctx \leftarrow$ CREATE_CONTEXT$(quote, paragraph)$   ▷ Creating context for a quote
14:                 $ctxid\_to\_text[ctxid] \leftarrow ctx$
15:                 $dataset$.append([$ctxid$, [pos_para_id], [candidate_id]])
16:                 $ctxid \leftarrow ctxid + 1$
17:                 if $quoteid$ in $quoteid\_to\_ctxid$.keys() then
18:                     $quoteid\_to\_ctxid[quoteid]$.append$(ctxid)$
19:                 else
20:                     $quoteid\_to\_ctxid[quoteid] \leftarrow [ctxid]$
21:                 end if
22:             end if
23:         end for
24:         save$(dataset)$
25:     end for
26: end for
27: save$(ctxid\_to\_text)$
28: save$(quoteid\_to\_ctxid)$
```

Algorithm 2 Create quote to book mapping

```
function CREATE_QUOTE_TO_BOOK_MAPPING(list_of_quotes, corpus_directory)
    Input: list_of_quotes, corpus_directory
    Output: quote_to_book_mapping
    quote_to_book_mapping ← {}
    for all quote in list_of_quotes do
        for all book_path in corpus_directory do
            if quote found in book_path then
                if quote in quote_to_book_mapping then
                    quote_to_book_mapping[quote].append(book_path)
                else
                    quote_to_book_mapping[quote] ← [book_path]
                end if
            end if
        end for
    end for
    return quote_to_book_mapping
end function
```

Algorithm 3 Segment book into paragraphs of fixed length

```
function SEGMENT_BOOK(text_document, paragraph_length)
    Input: text_document, paragraph_length
    Output: $paragraphs$
    $paragraphs \leftarrow \{\}$
    $current\_paragraph \leftarrow$ ""
    $current\_paragraph\_id \leftarrow 0$
    for $word$ in $text\_document.split()$ do
        $current\_paragraph \leftarrow current\_paragraph +$ "" $+ word$
        if len$(current\_paragraph) \geq$ paragraph_length then
            $paragraphs[current\_paragraph\_id] \leftarrow$ current_paragraph.strip()
            $current\_paragraph \leftarrow$ ""
            $current\_paragraph\_id \leftarrow current\_paragraph\_id + 1$
        end if
    end for
    if len$(current\_paragraph) > 0$ then
        $paragraphs[current\_paragraph\_id] \leftarrow$ current_paragraph.strip()
    end if
    return $paragraphs$
end function
```

Algorithm 4 Generate context for a quote in a paragraph

```
 1: function CREATE_CONTEXT(quote, paragraph)
 2:     Input: quote, paragraph
 3:     Output: context
 4:     context ← ""
 5:     quote_position ← paragraph.find(quote)
 6:     if quote_position $\neq -1$ then
 7:         preceding_40 ← paragraph[:quote_position].split(" ")[-40:]
 8:         following_40 ← paragraph[quote_position + len(quote):].split(" ")[:40]
 9:         context ← " ".join(preceding_40) " ".join(following_40)
10:     end if
11:     return context
12: end function
```

document, *chromaDB* as vector store. For fine-tuning the reranking models we use huggingface API [30].

FLAN-T5: We fine-tune our T5 models (base[31], large[32]) and *FLAN-T5-large*[33] with a learning rate of $2e^{-5}$ and a weight decay of 0.01 for a maximum of 10 epochs with a batch size of 4. We use a maximum of 1024 input tokens and one output token. Training T5 base, large, and Flan-T5-large take approximately 2, 5, and 6 hours overall, respectively, on a single RTX 4090 GPU. We use greedy decoding during inference and used *output_logits=True* while generating text to retrieve unprocessed probabilities assigned to a token. We use same hyperparameter setting for *Llama-3-8b-instruct*

bert-base: For fine-tuning *bert-base*[34] for the paragraph retrieval task, we search over a batch-size $\in \{4, 8, 16\}$, and set the learning rate of $2e^{-5}$. We set the maximum number of epochs to 10. We also perform a search over $n \in \{3, 6, 9, 12\}$ sampled negative paragraphs per positive paragraph for our paragraph ranking model. We select the best model using the dev set and the best paragraph model is trained with 9 negative examples and a batch size of 16. We used single NVIDIA Tesla P100 GPU for training the model.

Reader : For the span selection model (the multi-task and other transformers based baseline models), we cap the total length of the context and paragraph to 384 length wordpieces. In case the total length exceeds the maximum length (i.e., 384), we only truncate the paragraph. Similarly, for the quotable phrase identification model (i.e., the sequence classification model in the multi-task setting) we select a maximum length of 384. We fine-tune the publicly available *spanbert-large*[35], by setting the batch-size $\in \{4, 8\}$, learning rate of $2e^{-5}$. We fine-tune the model over 10 epochs and use early stopping based on the dev set. Again we used single NVIDIA Tesla P100 GPU for training the model. For the multi-task framework, the training process took 3.5 hours to complete. For the LLM inference we use single NVIDIA Tesla P100 GPU. Additionally, we applied 4bit quantization while loading the larger LLMs as those models would not fit in our GPU.

G Examples of paraphrased context

Table 8 shows one paraphrased example from QuoteR and Gandi dataset which were used as the target context. Quotus dataset having a separate target article, we did not require paraphrasing the context.

H Examples of LLM generated quotes

In Table 9 we provide examples of quotes extracted by different LLMs used in our experiments for a specific context and paragraph. We observe that, larger models (such as FLAN-T5-XL, Llama-3-8b) generate better quotes compared to the smaller models. However, Llama-3 merges some part of the context ("Sweet dewdrops") in the predicted quote. This is one of the precise reasons why standalone LLMs cannot be reliably used in the quote

text_splitter

[30] https://huggingface.co/
[31] https://huggingface.co/t5-base
[32] https://huggingface.co/t5-large
[33] https://huggingface.co/google/flan-t5-large
[34] https://huggingface.co/bert-base-uncased

[35] https://huggingface.co/SpanBERT/spanbert-large-cased

Dataset	Actual Context	Paraphrased Context
QuoteR	and for the great Peasant Revolt of 1381. John Ball's famous rhyme condensed the scorn for the nobles, the longing for just rule, and the resentment at oppression, of the peasants of that time and of all times:– " A hundred years after the Black Death the wages of a common English laborer–we have the highest authority for the statement–commanded twice the amount of the necessaries of life which could have been obtained for the wages paid under	For the significant Peasant Revolt of 1381, John Ball's renowned rhyme encapsulated disdain for the nobles, the yearning for fair governance, and resentment towards oppression. A century after the Black Death, the wages of an ordinary English laborer, as verified by the highest authority, were double the necessities obtainable with previous wages.
Gandhi	For, highest perfection is unattainable without highest restraint. Suffering is thus the badge of the human tribe. The goal ever recedes from us. The greater the progress, the greater the recognition of our unworthiness. Full effort is full victory. Therefore, though I realize more than ever how far I am from that goal, for me the Law of complete Love is the law of my being. Each time I fail, my effort shall be	The pinnacle of perfection requires the utmost restraint, and suffering becomes the emblem of the human experience. The goal remains elusive, and progress accentuates our sense of unworthiness. Full effort equates to complete victory. Despite realizing the vast distance from the goal, the Law of complete Love governs my existence. Each failure only strengthens my resolve.

Table 8: Examples of paraphrased contexts for QuoteR and Gandhi datasets.

extraction task, as it may augment the original text which could affect the sanctity of the quote.

Context	Paragraph	Actual quote	Used model	Extracted quote
Tonight, the heavens and the earth will mourn your passing, as sweet dewdrops fall from the sky. Your life is fleeting, and your time is coming to an end. The rose, with its	earth and sky, Sweet dews shall weep thy fall to-night, For thou must die. Sweet rose, whose hue, angry and brave, Bids the rash gazer wipe his eye, Thy root is ever in its grave, And thou must die. Sweet spring, full of sweet days and roses, a box where sweets compacted lie , My music shows you have your closes, And all must die. Only a sweet and virtuous soul, Like seasoned timber, never gives But when the whole world turns to coal, Then chiefly lives. George Herbert. THE LIE. Go, Soul,	Sweet spring, full of sweet days and roses, a box where sweets com-pacted lie	Bloomz-3b	earth and sky, Sweet dews shall weep thy
			FLAN-T5-large	rage, is ever in its grave
			FLAN-T5-XL	Sweet dews shall weep thy fall to-night, For thou must die. Sweet rose, whose hue, angry and brave, Bids the rash
			Falcon-7b	Sweet dewdrops fall from the sky
			Llama-3-8b	Sweet dewdrops fall from the sky. For thou must die. Sweet rose, whose hue, angry and brave, Bids the rash gazer wipe his eye, Thy root is ever in

Table 9: Quotes extracted by different LLMs used for a specific context and paragraph