ConTAXt Retrieval for Long-Form Question-Answering

Stanford CS224N Custom Project

Will Roberts
Department of Computer Science
Stanford University
robertsw@stanford.edu

Usman Iqbal Hanif
Department of Computer Science
Stanford University
uhanif@stanford.edu

Winston Shum
Department of Economics
Stanford University
wshum@stanford.edu

Abstract

We explore the application of two long-form question-answering (LFQA) paradigms to the ominous and equally convoluted field of taxes. We first scrape a text knowledge corpus from the federal tax code, IRS materials, and online tax guides. We then distill from said corpus a set of 10,000 pertinent questions whose answers are found in the text. Using these materials, we finetune (1) A Generative Pre-trained Transformer model (GPT-2) and (2) a Retrieval-Augmented Generation (RAG) model, assessing their performance with the aid of a certified public accountant. Our goal is to build an LFQA tool that empowers the average American taxpayer.

1 Key Information to include

No external collaborators, mentors, or sharing projects.

2 Introduction

Tax season can be daunting for many Americans, as it involves dealing with complex and inscrutable tax codes and regulations. In fact, according to a recent survey, Americans spend on average anywhere between 11-13 hours preparing and filing their tax returns every year (FreshBooks, 2023).

Not only are tax codes lengthy and convoluted—they also change frequently. This presents a challenge for people looking to stay informed with the latest tax laws, regulations, and loopholes. As a result, many opt to instead hire expensive accountants, which can be costly and still time consuming.

Most tax education software available today fails to capture the intricacies and nuances of how certain tax laws apply to an individual's situation. Advances in Natural Language Processing (NLP) techniques expose an opening in the market for an intelligent tax question-answering system, one that can provide more customizable and robust tax education services.

In this research report, we aim to develop a question-answering system that uses Natural Language Processing techniques to provide accurate and reliable tax advice. Our system aims to go beyond the current generation of models and provide personalized tax advice that is tailored to the individual's specific situation. By doing so, we hope to bridge the gap between the complexity of tax codes and the general public's understanding of them. Our proposed system intimates time- and money-saving potential for individuals and businesses, arming them with greater confidence as they head into tax season.

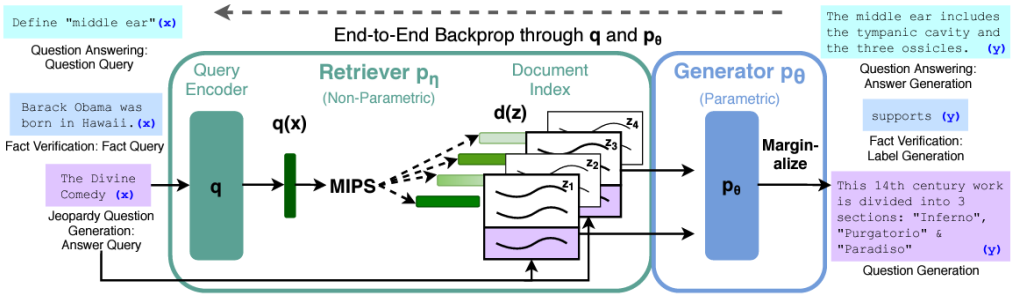
3 Related Work

Given our goal to answer tax-related questions, we explore two types of models: (1) Autoregressive Transformer models such as GPT and (2) Retrieval-Augmented Generation (RAG) models that use document retrieval in conjunction with a generative model.

Long-Form Question-Answering Long-form question-answering (LFQA) is a special variant of open-book abstractive QA. The term is first presented by a group of Facebook researchers in 2019 (Fan et al., 2019). Described as "a task requiring elaborate and in-depth answers to open-ended questions," LFQA is realized through a dataset comprised of 270K threads from a Reddit forum titled "Explain Like I'm Five" (ELI5). Compared to existing datasets, ELI5 contains diverse questions requiring multi-sentence answers. In the tax industry, where complex laws can take on numerous interpretations and often require substantial explanation and simplification, we hypothesize that a pre-trained ELI5 model is an ideal component of our project.

Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks (RAG) This paper on Retrieval-Augmented Generation presents a general RAG model that finetunes a pre-trained parametric language model and excels in knowledge-intensive tasks (Lewis et al., 2020). The RAG model combines both pre-trained parametric and non-parametric memory for language generation tasks, generating more detailed and accurate texts than parametric-only models. The paper describes a RAG model that uses a Dense Passage Retrieval (DPR) model, which is capable of retrieving top-k documents and building a document index using a double-encoder architecture. After concatenating an input question and the retrieved documents, the generator component is then modeled with BART, a bidirectional autoregressive transformer that combines BERT's bidirectional encoder and GPT's autoregressive decoder. The RAG model then marginalizes over seq2seq predictions with various documents. See the figure below to better understand the RAG model. After joint training of the retriever and generator components, open-domain RAG models outperform previous language generation approaches in knowledge-intensive tasks.

Figure 1: RAG Model Approach (Lewis et al., 2020)



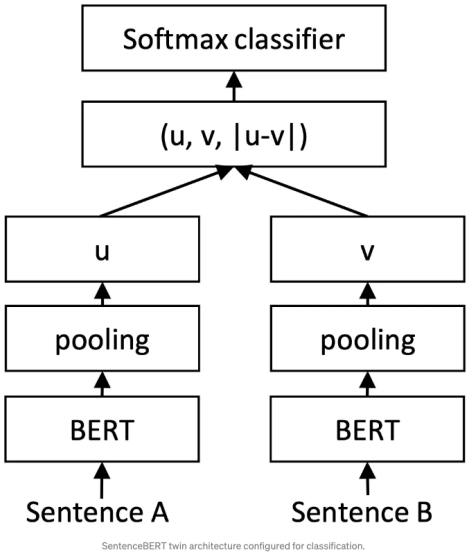
Given the success of the RAG model with knowledge-intensive tasks, we reference the approach presented in the paper as a likely solution to our goal, mimicking the retriever and generation components of the RAG model. In deciding which encoding or retrieval model to use, we consider Dense Passage Retrieval, BM25, Universal Sentence Encoder and S-BERT, but refine our experiments to the latter two for the scope of this project.

Universal Sentence Encoder (USE) Google's Universal Sentence Encoder encodes sentences into 512-dimensional embedding vectors that can be used in a variety of NLP tasks (Cer et al., 2018) While previous language models use pre-trained word embeddings such as word2vec or GloVe, Universal Sentence Encoder enables the creation of sentence embeddings that outperform word embeddings in many transfer task performances. The Universal Sentence Encoder presented in the paper also has two different encoding models, one that uses a transformer-based sentence encoding and another that uses a Deep Averaging Network (DAN). The transformer-based sentence encoding uses the encoding sub-graph of the transformer architecture, while the DAN averages embeddings for words and passes them through a feedforward deep neural network. For our purposes, the general Universal Sentence Encoder is a potential option as an encoding and retrieval model, as it can be used to encode both

queries and documents and is referenced as a baseline for comparing our finetuned S-BERT-based retrieval model, explained in the following subsection.

Sentence BERT: Sentence Embeddings using Siamese BERT-Networks (S-BERT) While models based on DPR, BM25, and Universal Sentence Encoder are all options for our retrieval model and can be used on tasks such as open-domain QA, the paper on S-BERT provides a model to learn semantically meaningful sentence encodings by modifying the standard BERT with a siamese network architecture, allowing fixed-size vectors to be created for entire input sentences (Reimers and Gurevych, 2019). S-BERT's siamese network architecture relies on two BERT networks with mirrored weights. Each BERT sub-unit generates word encodings for separate sentences, then employs a pooling operation (either mean or max pooling) to produce a single embedding for each sentence. The resulting semantically meaningful sentence embeddings can be compared with functions such as cosine-similarity or softmax. Importantly, the S-BERT model greatly improves computational speed compared to other BERT sentence encoders and can outperform the Universal Sentence Encoder on Semantic Textual Similarity tasks. For our purposes of generating answers for tax-related questions, which require a detailed understanding of an industry-specific knowledge base, we utilize the S-BERT model's ability to learn sentence embeddings by finetuning it to the tax domain.

Figure 2: SentenceBERT with a Siamese Network



4 Approach

Compiling a Segment-Specific Knowledge Base We invest significant time in amassing and processing a large text knowledge base surrounding tax laws and certified advisory materials, we explain these in depth in Section 5.

Approaches to Long-Form Question-Answering (LFQA) When given a large knowledge database, there are two primary approaches we can take to answer questions in an LFQA manner.

The first involves fine-tuning an autoregressive language model on text data covering the topics in question. This approach is effective when the knowledge base is relatively consistent and the questions being asked fall within a specific domain. In the case of tax codes and guides, we surmise that the preceding qualities held true, hence, we choose this as our first approach. Additionally, this approach requires little to no data labeling, as this class of LLMs can train on raw text alone. By training a generative model on our knowledge base, it can gain a better understanding of the language and terminology used in that domain, making it more fluent in answering questions.

The second approach we attempt is Retrieval Augmented Generation (RAG), which incorporates an information retrieval component that is then fed into the prompt for a generative component. This allows the model to access and integrate information from the knowledge base into its generated responses, ideally resulting in more accurate and informative answers. While standalone generative models can be finetuned on knowledge bases, they easily lose accuracy or factual correctness in their responses—despite sounding correct. RAG overcomes this by retrieving the original knowledge passages and presenting them to a generative model at the point-of-prompt rather than relying on parametric knowledge recall from training.

4.1 Autoregressive

4.1.1 GPT-2 Baseline

For our baseline model, we deploy Huggingface's GPT-2 medium model (355M parameters) without any task-specific fine-tuning (Radford et al., 2019). GPT-2 uses a transformer architecture that is composed of only decoder blocks. Every new word is predicted by using the prior word and self-attention (for contextual dependencies). It is trained on an industrial-scale web-scraped dataset using unsupervised learning. This is an intuitive baseline since we plan on finetuning and modifying our GPT-2 based model under the context of tax-related questions and information.

4.1.2 Finetune GPT-2

Our first major step is finetuning GPT-2 over our tax corpus. We make sure to clean our text corpus to remove potentially errant formatting characters. We then retrain the HuggingFace out-of-the-box GPT-2 model on the unlabeled corpus text with the ultimate hope that the model will adopt tax-specific language to more thoroughly and knowledgeably answer questions. The performance of our finetuned GPT-2 model surpassed that of our baseline GPT-2 model. We'll further address these results in the Experiments and Analysis sections below.

4.2 RAG

While finetuning GPT-2 shows promise, there is concern that purely autoregressive models often lack correctness in fact or logic-intensive responses. Retrieval Augmented Generation offers a promising solution. RAG models retrieve relevant information from the broader knowledge base and incorporate that into a generator's prompt. In effect, this leads to better knowledge-base preservation, as we retrieve and prompt content, contrary to just training on it. RAG models therefore allow for a more closely guided generator output.

As Figure 2 shows, we parse our corpus and generate embeddings sentence-by-sentence. The tax-related query is also encoded, and the corpus sentences with the highest cosine similarity to the query vector are then prompted to the generative model, along with the query itself. The following subsections explore the encoder and BART models in greater detail.

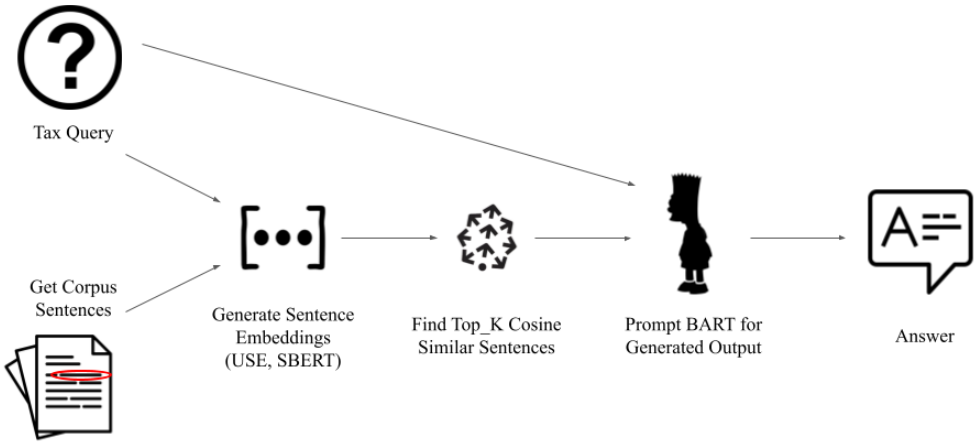
4.2.1 USE + BART

We first utilize Google's Universal Sentence Encoder to generate 512-dimensional sentence embeddings which are optimized to encapsulate the semantic meaning of the text. USE is trained on a variety of sources—including web pages, news articles, and books—and is known for it's ability to encode text of any length without truncation. While this may seem a convenient solution for large text body encoding (as opposed to sentence-level encoding), embeddings for larger texts undergo significant dilution. This effect makes text comparison of varying lengths—such as a question and paragraph—quite unstable and thus inspires alternative approaches to encoding.

4.2.2 S-BERT (sentence encoder)+ BART

As mentioned in Section 3, BERT typically operates on a word-by-word basis, making it inefficient and ineffective at capturing semantic meaning of sentences. Sentence-BERT (S-BERT) addresses this issue, allowing us to compute dimension 768 embeddings over a maximum of 128 tokens, which is sufficient for most sentences. The Siamese BERT network can then draw comparisons between the two sentences. We use S-BERT to determine which sentences in our corpus are most cosine similar to the query, pulling top-k matches to be fed into a BART generator via prompting.

Figure 3: Our RAG Approach



4.2.3 S-BERT (sentence encoder) fine tuned + BART

Finally, we repeat the same step from above, this time finetuning our S-Bert model on question-answer data which we generate. The open-source nature of S-BERT models certainly provides an advantage to Google's Universal Sentence Encoder, which is notoriously difficult to finetune and lacks any substantive developer community.

5 Experiments

5.1 Data

Our retrieval documents are harvested from myriad sources:

1. The entire United States Federal Code Title 26: Internal Revenue Code is scraped, collated, and summarized into continuous, digestible prose using OpenAI's GPT 3.5-turbo API. The resulting output constitutes over 3.5 megabytes of text (OpenAI, 2020; Office of the Law Revision Counsel of the United States House of Representatives, 2023).
2. The IRS Website hosts a number of resources to help Americans file their tax returns, the most bountiful of which include (1) a large collection of FAQs with answers and (2) tax guides for each major section of tax law (Service, 2023). We scrape and process both of these sources into raw text files for finetuning. Together, these sources comprise 6.5 megabytes of text.
3. Beyond the IRS, there exist a number of commercial tax guides, all of which are available online. We download, convert, and use the following guides in our training processes:
    - *Lower Your Taxes - BIG TIME! 2023-2024 Small Business Wealth Building and Tax Reduction Secrets from an IRS Insider* Botkin (2022),
    - *Financial independence (getting to point X) a comprehensive tax-smart wealth management guide* Vento and Mackay (2018),
    - *J.K. Lasser's 1001 Deductions and Tax Breaks 2022 Your Complete Guide to Everything Deductible* Weltman (2021),
    - *The Book on Advanced Tax Strategies Cracking the Code for Savvy Real Estate Investors* Han and MacFarland (2020),

These materials are then concatenated into a single text corpus representing our knowledge base. In total, for our context documents, we have over 100,000 sentences at our disposal for fine-tuning and

evaluation. While raw, unlabeled text is sufficient for fine-tuning an autoregressive model such as GPT-2, QA-formatted data is required for our case of effective passage retrieval. Once more using GPT-3.5 Turbo, we are able to successfully generate 10,000 question-answer pairs by segmenting our corpus into 5-sentence sections and running the following script:

```
prompt = """Given the following context, generate:
(1) a tax-related question whose answer can be found in the context
(2) The particular sentence that best contains the answer within the
context, verbatim.

Output all of this like you would declare a python list (with brackets)
of strings in this particular order. The list should only contain
these two elements. \n context: """

try:
    response = openai.ChatCompletion.create(
        model="gpt-3.5-turbo",
        messages=[
           {"role": "user", "content": prompt+section},
        ],
    temperature = .4,
    top_p=1
)
except: continue
```

We split this labeled data into training and evaluation sets for retrieval finetuning and evaluation.

5.2 Experimental details

5.2.1 Autoregressive

We finetune HuggingFace's GPT-2 medium model using the raw text of our knowledge corpus with a batch size of 16 over 5 training epochs. For generation, we set temperature to default (0.7), top_k to 50, top_p to 1, and do_sample to true.

5.2.2 RAG

We operate our retrieval process on a sentence-by-sentence level, generating embeddings for each sentence within our knowledge corpus and uploading to a Pinecone S1 index for easy retrieval. For each question posed, the model generates query embeddings and requests a top-k number of sentences from Pinecone, each of which ideally contains relevant corpus text. Since answers to questions can exist over multiple sentences, we concatenate the surrounding 4 sentences within the context to each of these top-k Pinecone results. This technique allows for larger passage retrieval despite only having sentence-level training data. We also notice that S-BERT and USE both work optimally when the two texts being compared are of similar lengths, so this approach should optimize retrieval performance.

Then, as mentioned in Section 4, we structure a query consisting of the original question—followed by all relevant context passages—and feed that to the BART model for answer generation.

Our pre-trained S-BERT model is trained on Microsoft's mpnet-base model and fine-tuned on a 1 billion sentence pair dataset(Reimers, 2021). We finetune this model on pairs of sentences comprised of the generated question with its corresponding answer sentence in the corpus. 8400 of these pairs were randomly selected for training, while 1582 were reserved for evaluation. We train with a batch size of 64 spanning 5 epochs with the number of warmup steps equal to the size of the training data multiplied by the number of epochs multiplied by 0.1 (representing 10% of the training data). Since our data lacks a cosine similarity label, we use Multiple Negative Ranking Loss as our loss function, which is quite effective for training datasets consisting of positively-related pairs. We did not finetune BART, the generative part of our RAG structure. We use a HuggingFace model pre-trained on ELI5(Jernite, 2021) and find its performance to be more than sufficient. With finetuning complete, we then set out to evaluate the performance of our 5 models.

5.3 Evaluation method

First, we quantitatively evaluate the retrieval components of our RAG models: S-BERT, Finetuned S-BERT, and USE, using our evaluation set. Before any querying, each model generates an encoding for every corpus sentence and uploads them to Pinecone. When a question is presented to the model, it's embedding is sent to Pinecone, and the top-k most similar context sentences are returned in order of decreasing similarity. We assess for, within a top-k of 20 results, how many times a question's correct answer sentence appears. These question-answer pairs come from our evaluation set of size 1528. Additionally, as a sanity check, we evaluate 500 random question-answer pairs from our training set to gauge Finetuned S-BERT's training retention relative to out-of-the-box models. At the very least, we expect Finetuned S-BERT to outperform on this set of questions. See Figure 4 for results.

We further assess with what level of success these models find the correct answer sentence within a top-k of 5, and we weigh the ranking in which they are returned. We score each position using a decay factor of 2 so that the similarity results earn the following points in order from first to fifth: 100, 50, 25, 12.5, 6.25. See Figure 5 for results.

To assess the correctness of the generative models, we composed a test of 50 tax-related questions. 25 of these questions are pulled from our evaluation set, and the following 25 are supplied to us by a certified public accountant. We then query all five models: GPT-2, GPT-2 Finetuned, S-BERT+BART, S-BERT Finetuned+BART, and USE + BART, and submit the results to the CPA to assess. We (kindly) ask the CPA to rank each model for each question based first on correctness then on clearness in the case of any draws. See Figure 6 for results.

Figure 4: Observation counts of correct context sentence within the top 20 model results.

Model Counts Within Top_K=20			
	S-BERT Finetuned	S-BERT	USE
Evaluation Set	1,461 (92%)	1,380 (87%)	1,131 (71%)
Training Set	477 (95%)	445 (89%)	368 (73%)
Total	1,938 (93%)	1,825 (87%)	1,499 (71%)

Figure 5: Position-weighted score of correct context sentence observed within the top 5 model results

Model Scores for Decayed-Position Top_K=5				
	S-BERT Finetuned	S-BERT	USE	Maximum Possible
Evaluation Set	107,726	98,180	75,725	152,800
Training Set	34,502	31,415	24,997	50,000
Total	142,228	129,595	100,722	202,800

Figure 6: Frequency of ranks by model, as scored by a certified public accountant. 50 questions total.

CPA Ranked Responses

	S-BERT FT/BART	S-BERT/BART	USE/BART	GPT2 FT	GPT2	Total
1st	19	10	8	10	3	50
2nd	13	16	9	5	7	50
3rd	10	13	12	6	9	50
4th	6	8	15	11	10	50
5th	2	3	6	18	21	50
Total	50	50	50	50	50	

6 Analysis

For retrieval, our finetuning proves successful in augmenting S-BERT's comprehension of tax-specific language. The finetuned model is able to recognize specific sections and form numbers, whereas the other two tend to lose them entirely. Furthermore, the non-finetuned models are overly sensitive to commonly occurring words in the corpus such as "tax" or "deduction," often neglecting other, more unique keywords within the question text. This behavior leads to more semantically irrelevant results.

RAG models outperform their purely autoregressive counterparts when it comes to assessing correctness and overall quality of generated response. One factor that may contribute to the poor performance of the finetuned GPT-2 FT's model is the sheer variety of structure, content, and tone encompassed within our training corpus. We initially assume that the corpus text was uniform, but later discover that the tone and person of the commercial tax guides varies dramatically from that of the IRS. As such, it is not uncommon for GPT-2 FT's output to read as disjointed or hard to follow on a sentence-by-sentence basis. Further, while the majority of our training corpus is easy to read, certain documents (particularly those scraped from the IRS) contain complex language that assumes fairly extensive reader knowledge. We theorize that these documents often contribute to densely generated language. We further theorize BART is able to cut through this more successfully due to its ELI5 dataset pre-training. Note also, as Figure 6 shows, GPT-FT tends to perform at the ranking extremes.

7 Conclusion

We set out to use advanced NLP techniques to help the average American better understand the complicated and lengthy tax code. We attempt five different approaches, a handful of which incorporate current state-of-the-art models. Next steps include finetuning BART for generation and incorporating a bag-of-words algorithm like Okapi BM25.

It's also important to note that all models' performance still fall short of a certified accountant's for the majority of questions, and significant improvement remains in order to achieve market viability.

References

Giambattista Amati. 2009. Bm25. In Ling Liu and M. Tamer Özsu, editors, *Encyclopedia of Database Systems*, pages 263–264. Springer.

Vladimir Blagojevic. 2021. bart-lfqa.

Vladimir Blagojevic. 2022. Long-form qa beyond eli5: An updated dataset and approach.

Sandy Botkin. 2022. *Lower Your Taxes - BIG TIME! 2023-2024: Small Business Wealth Building and Tax Reduction Secrets from an IRS Insider*, 9th edition. McGraw Hill.

James Briggs. 2020. Generative question answering with long-term memory.

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nic Limtiaco, Rahul John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, et al. 2018. Universal sentence encoder. *arXiv preprint arXiv:1803.11175*.

Kim Changyeop. 2020. How to fine-tune gpt-2 for beginners.

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

Omar Espejel. 2022. Train and fine-tune sentence transformer models.

Angela Fan, Yacine Jernite, Ethan Perez, David Grangier, Jason Weston, and Michael Auli. 2019. Eli5: Long form question answering.

FreshBooks. 2023. How long does it take to prepare a tax return?

Amanda Han and Matthew MacFarland. 2020. *The Book on Advanced Tax Strategies: Cracking the Code for Savvy Real Estate Investors (Tax Strategies, 2)*. BiggerPockets.

Yacine Jernite. 2021. Explain anything like i'm five: A model for open domain long form question answering.

Viktor Karlsson. 2020. Sentencebert–semantically meaningful sentence embeddings the right way.

Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen tau Yih. 2020. Dense passage retrieval for open-domain question answering. *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6775.

Patrick Lewis, Ethan Perez, Aleksandr Pitkus, Fabio Petroni, Vladimir Karphukhin, Naman Goyal, Heinrich Kütler, Mike Lewis, Wen tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *arXiv preprint arXiv:2005.11401*.

Sambit Mahapatra. 2019. Use cases of google's universal sentence encoder in production.

Prakhar Mishra. 2022. Understanding dense passage retrieval (dpr) system.

Office of the Law Revision Counsel of the United States House of Representatives. 2023. Title 26-Internal Revenue Code.

OpenAI. 2020. Gpt-3.5-turbo api endpoint.

Pinecone. 2021. Abstractive question answering.

Pinecone. 2022. Sentence transformers: Meanings in disguise.

ProjectPro. 2021. Transformers bart model explained for text summarization.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Nils Reimers. 2021. flax-sentence-embedings/all-datasets-v3-mpnet-base.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *EMNLP-IJCNLP 2019*, pages 3982–3992.

Nicholas Renotte. 2021. Generate blog posts with gpt2 & hugging face transformers | ai text generation gpt2-large.

Internal Revenue Service. 2023. Irs publications.

Francois St-Amant. 2019. How to fine-tune gpt-2 for text generation.

John J. Vento and Todd Mackay. 2018. *Financial Independence (Getting to Point X): A Comprehensive Tax-Smart Wealth Management Guide*, 2nd edition. Wiley.

Barbara Weltman. 2021. *J.K. Lasser's 1001 Deductions and Tax Breaks 2022: Your Complete Guide to Everything Deductible*, 2nd edition. Wiley.