Zakery Clarke

*Candidate*

Computer Science

*Department*

This thesis is approved, and it is acceptable in quality and form for publication:

*Approved by the Thesis Committee:*

Lydia Tapia , Chairperson

George Luger

Leah Buechley

**by**

THESIS

Submitted in Partial Fulfillment of the
Requirements for the Degree of

The University of New Mexico
Albuquerque, New Mexico

**LEARNING INTERMEDIATE REPRESENTATIONS FOR QUESTION ANSWERING SYSTEMS**

by

Zakery Clarke

B.S., Computer Science, University of New Mexico
M.S., Computer Science, University of New Mexico

## ABSTRACT

Question answering systems are models that can perform natural language processing (NLP) on a question, retrieve an answer from a datasource, and communicate it to a user. In question answering systems, it is important for the system to learn an underlying representation for a piece of text. There are many systems that have achieved incredible accuracy on question answering datasets such as the Stanford Question and Answer Dataset (SQuAD), but these systems often encode their knowledge in a manner that is impossible to verify. Many current models would benefit more from verifiability, than marginal accuracy improvements.

We propose a method to learn representations for a piece of text in a manner that is human-auditable. The model accomplishes these goals by leveraging the power of modern transformer neural network models and a unique dataset to create a model that is accurate and interpretable.

## Table of Contents

## Introduction

The large amount of textual data along with innovative neural network architectures has led to incredibly effective question answering systems **[15]**. These systems are trained on large corpora  such as WikiData **[12]** and SQuAD **[8]** and are able to learn accurate underlying representations for these articles of text **[15]**. Many of these systems have even started to surpass human performance on datasets such as SQuAD **[16]**. Models can accurately summarize information and extract answers from large pieces of text, which can aid experts in information retrieval and help reinforce decision making, without having to examine thousands of pages of documents. However, it is important to determine the level of understanding that a given model has about a piece of text, to ensure that answers are accurate.

Natural language understanding is an important task that transforms human readable language into machine readable forms. These representations allow machines to reason about complex concepts. There are many forms of intermediate representations that have their uses in different contexts. One such form is subject, verb, object  (SVO) knowledge graph tuples, which represent simple relationships and can be matched to answer queries.

By translating natural language into an intermediate representation, machines can define their knowledge in a verifiable manner. Forcing the model to learn an

intermediate representation may help improve generalization and prevent overfitting, as the model cannot simply map questions to answers.  This is important to ensure that a model will extrapolate well to new pieces of text. Intermediate representations also improve the interpretability of a model, as a human expert can examine the knowledge graph to ensure its accuracy. This has vast applications in many industries that rely on highly accurate inferences for decision making. Many of these industries can not tolerate errors, as major funding and even lives can be at risk. By having a human verifiable representation, we can reduce the risk of the predictions of a model and can better diagnose errors that the model may be generating.

### Our Work

We are interested in developing a model that can retain super-human performance, while also improving interpretability. This model could be used in many fields to generate knowledge structures about large pieces of text. The insights generated from this can be used to guide decision making, build chatbots and other related NLP tasks. The model needs to be easily verifiable by a human expert to ensure that the system can function correctly in high risk environments.

There have been many methods used to generate machine representations for question and answering tasks **[15]**. In this paper we build upon existing methods

to create a model that can infer SVO graph tuples from an article of text. This model is interpretable, and uses a transformer architecture trained on the SQuAD dataset to generate novel knowledge graphs that have not been annotated. These inferred knowledge graphs improve on existing methods by translating sentences into a machine readable and human understandable format that describes the knowledge of a passage of text.

## Background

### Question Answering Systems

Many papers have explored question answering systems. Most state of the art question answering systems use the transformer network architecture to learn an underlying representation for a piece of text **[6]**. These networks have had great success, and have even exceeded human performance on SQuAD **[8]**. However, these models are not interpretable. The inner workings and the knowledge that these transformers are able to learn from a passage of text only exist in the encoding of the system. It would be helpful to have a system that is able to specify, in human-readable terms, what its domain knowledge about a topic is, and how it generates an answer to a question.

Attempts have been made to better interpret the encodings of these models. The attention mechanism used by most state of the art question answering models allows us to view what the model deemed important during a given task **[20]**. **Fig 1** shows an example of the tokens a model was paying attention to while generating the answer for a given question **[21]**. This can help improve the interpretability of a model, as we can identify keywords that the model thought was relevant for a given question.

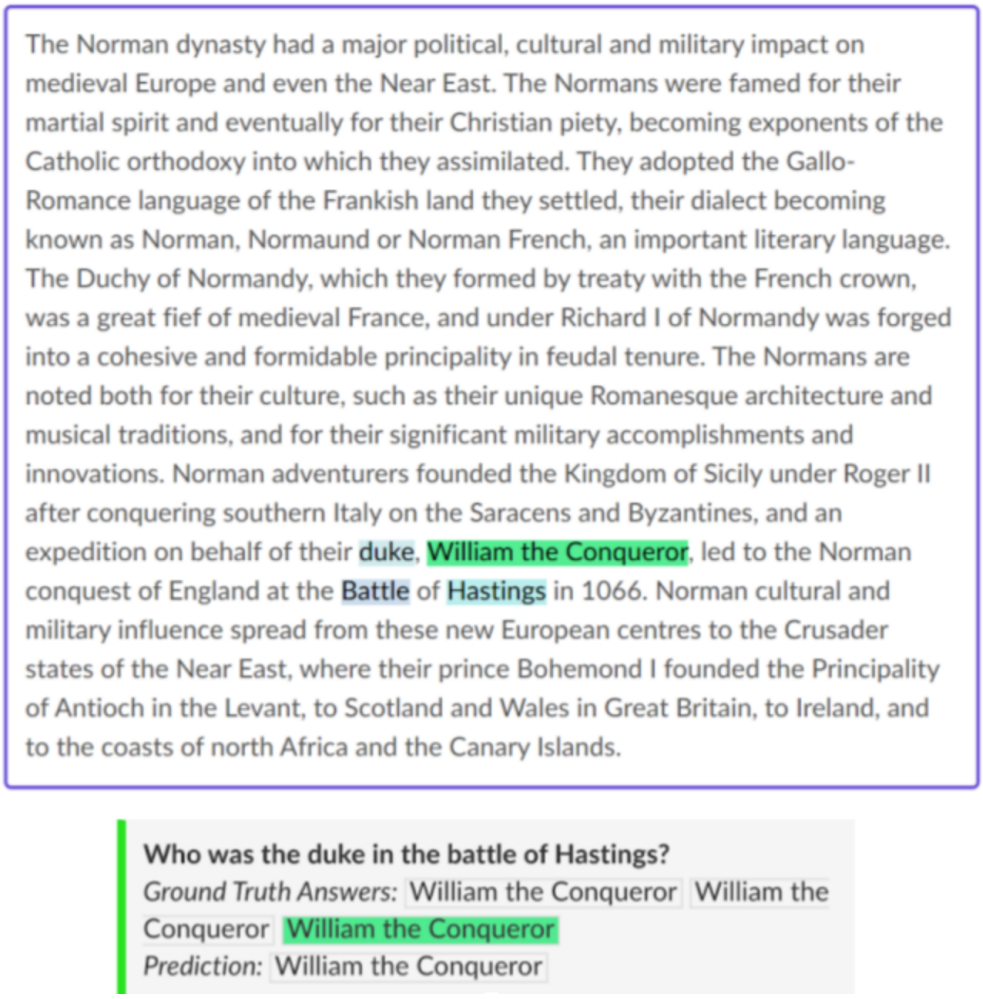


**Fig 1.** Attention Mechanism of a model on a question answering task **[21]**

This attention mechanism doesn't allow full interpretability however. The model is limited in its expressiveness, and can only identify relevant keywords instead of generating a complete representation of the knowledge of the model. Furthermore, the model is dependent on the question in order to be able to communicate its knowledge about a passage of text. This also prevents the model from being corrected if a faulty representation is found. An intermediate

representation would allow the model's knowledge to be edited by a human expert.

### Intermediate Representations for Natural Language

There are many representations that have been created to encapsulate the knowledge of a system. All representations have a tradeoff between specificity and generalization. Some systems, such as conceptual dependency theory describe scenarios in detail, but are unable to generalize to similar scenarios **[14]**. Other systems are too general and will match queries which are too similar to many other representations **[14]**. A balance of specificity and generality is needed to create an adaptive system that encapsulates the semantics of a sentence.

### Graph Representations

Graphs are a common structure for intermediate representations, as they are flexible and can describe complicated relationships. The graph representations range from the complete conceptual dependency theory to key value data such as DBpedia.

### Conceptual Dependency Theory

Conceptual dependency theory is a formal architecture to describe relationships between objects and actions that act upon them **[14]**. This formalism is complete

at describing situations, but is often too pedantic to be of use and does not generalize well. Conceptual dependency theory represents the relationships between objects using specified actions and modifiers. This allows for complex interactions to be described in a formal manner. **Fig 2** demonstrates some sample use cases along with their corresponding conceptual dependency graphs.

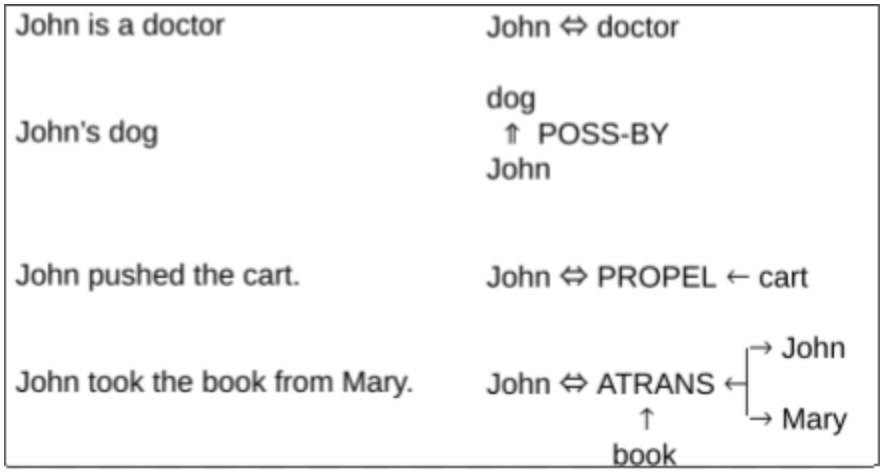


**Fig 2.** Selected manually parsed conceptual dependency graphs for a given sentence adapted from "Cognitive Science" **[13]**

This structure supplies a formal and complete representation of the semantics of a sentence, but fails to extrapolate accurately to new data. Conceptual dependency parsers are able to create a specific representation of a sentence, but are often difficult to generalize and need manual annotation.

### Concept Graphs

Concept Graphs offer a less strict paradigm, where each node can be a learned concept. As shown in **Fig 3** each node consists of a concept that is connected to other nodes. Concept graphs are also able to represent classes and subtypes, such as in **Fig 3**, where “Fido” is a specific instance of “dog”. This allows for generalization and even allows concept graphs to be directly translated into propositional logic.

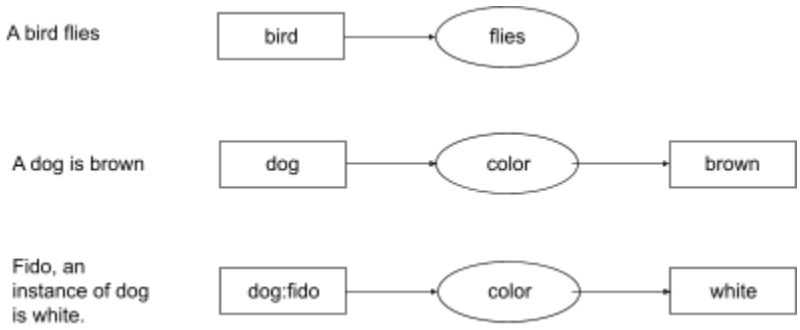


**Fig 3.** Selected Conceptual graphs adapted from “Artificial Intelligence” **[14]**

### Structured Knowledge Graphs

Other less formal graph representations exist, such as knowledge graphs **[10]**. Knowledge graphs consist of concepts located at the nodes, with some relationship between concepts along the edges. These simple relationships can

be used to describe abstract concepts in a more accessible way. For example, DBpedia is a large knowledge graph resource that contains manually annotated objects along with their respective properties **[10]**. This knowledge graph is generated from curated data in Wikipedia articles, including the infobox information. Another knowledge graph system is Wikidata, which relies on users to manually annotate relationships between concepts. Shown in **Fig 4** is a selected entry from Wikidata, which demonstrates sample relationships between separate Wikipedia pages. These curated databases of knowledge are a helpful backend for an NLP system, but require manual annotation from a human to function properly, and cannot generalize to novel pieces of text.

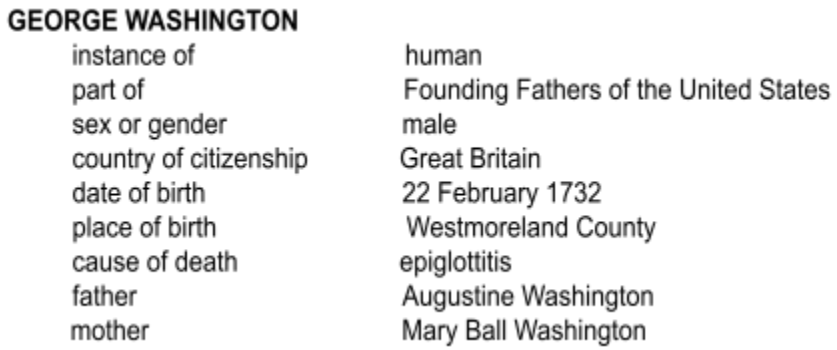


**Fig 4.** Selected relationships for entry “George Washington” from Wikidata dataset **[12]**

### Logical Forms

Logical representations have also been popular, such as (Luke S. Zettlemoyer et. al,2005), which parsed queries into a  lambda calculus grammar to evaluate in a programming environment **[11]**. Lambda calculus is Turing complete, so it can describe complicated concepts through the use of variables and function evaluation. (Hrituraj Singh, et al, 2020) used first order logic as an intermediate representation which could be unified to yield answers to a given question, albeit with a significant loss in accuracy **[2]**. First order logic can be evaluated in a logic programming language such as Prolog to attempt to unify and yield an answer. Both of these methods aided machine interpretation, but often can be ambiguous when representing the semantics of a piece of text.

### Learned Representations

The schema used to represent natural language is important, but essentially useless if we are unable to automatically generate the representation on novel pieces of data. Ideally, we require a system that can parse any piece of text into a consistent, coherent and complete description of a passage of text in a machine readable form. There has been exploration into using hand built parsers for natural language understanding tasks, but it is hard to develop a system that can parse all of the rich semantics of language. Therefore, we look to systems

that are able to learn a representation for a piece of text, without an explicit description of the method used to generate the representation.

### First Order Logic

Previous work on learned representations has been done to create more interpretable NLP systems. (Hrituraj Singh, et al, 2020) trained a Long Short Term Memory (LSTM) network to generate novel First Order Logic (FOL) pairs from a passage of text. The LSTM network used the Stanford Natural Language Inference (SNLI) dataset as input, and outputted FOL sequences. **Fig 5** shows a sample mapping from a sentence to first order logic that can be unified to evaluate a natural language expression.

"All humans eat"

$$\forall A(\exists B(human(A) \wedge eat(B) \wedge agent(B, A)))$$

**Fig 5.** Sample mapping from natural language to first order logic **[1]**

These were then evaluated using a unification algorithm to determine if one sequence could be used to infer the other. This paper had mixed results, partially due to a suboptimal dataset, with a significant loss in accuracy in exchange for a more interpretable representation.

### Learned Graph Representations

Graph structures have also been dynamically generated for intermediate representations. For example, a simple parser was used for geographical datasets through the use of querying the geobase dataset **[5]**. This system used a manually annotated dataset that mapped questions to database queries using an internal structure related to the problem domain. The results of this method were accurate, but the parser was required to have specific domain knowledge, which prevents generalization to new datasets. (Lorand Dali, et al, 2008) explores parsing sentences into SVO tuples using a support vector machine (SVM) **[7]**. SVO encapsulates simple sentences and generalizes well, however, the data has to be manually curated and annotated, and can fail to represent complex relationships. Learned representations allow computers to better represent their knowledge of a domain in a verifiable manner, but often require large manually annotated datasets.

## Methods & Experiment

In order to improve upon these methods, we need the ability to answer questions from an intermediate representation. Unfortunately, there are few data sources that contain articles of text along with a machine friendly interpretation of the article.

### Intermediate Representation

We decided that generating a SVO knowledge graph from a passage of text would be the best method, as it strikes a balance between generalization and specificity. This knowledge representation functions well in question answering systems, as questions often have a single subject and verb, along with an answer for the object. The simple triplet form is also easier for a neural network to generate, as the rules for the grammar are simply a tuple of the form (subject, verb, object).

### Dataset Generation

There are no datasets that contain SVO knowledge graphs paired with an article of text. The SQuAD dataset consists of a passage of text, along with questions and answers about the text. A key property of the SQuAD dataset is that the answer to any given question must be referenced in the passage of text that accompanies it. We can leverage this property to generate our intermediate

representation. We use the spaCy **[9]** dependency parser to parse a given question into a (subject, verb, object) tuple, with either the subject or object filled in with the answer to the question, as demonstrated in **Fig 6**.

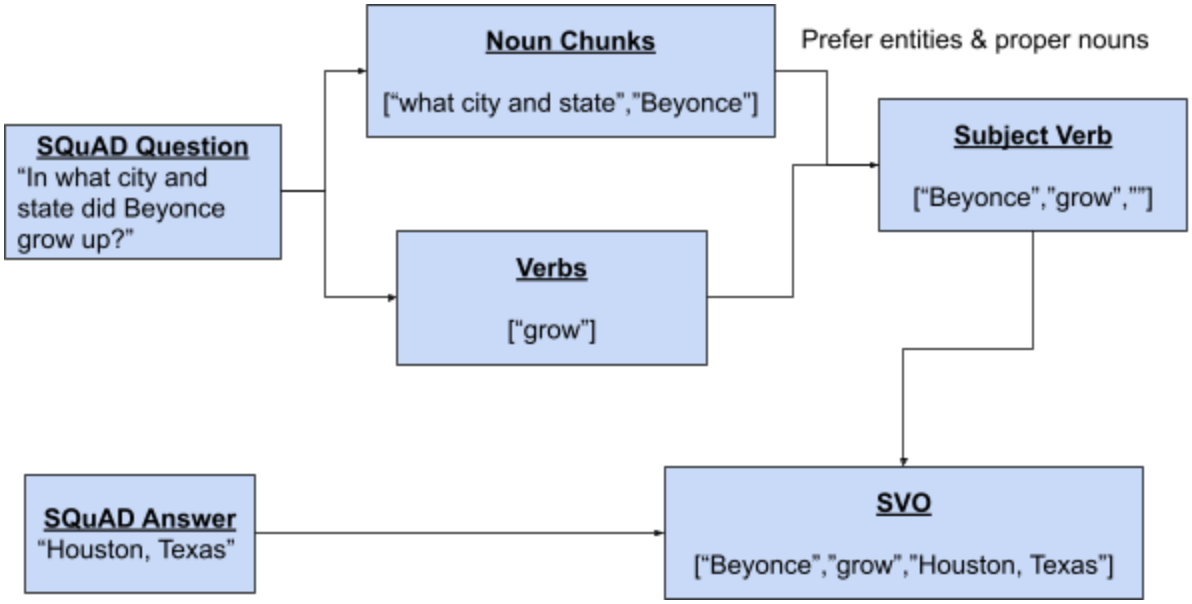


**Fig 6.** Example translation from SQuAD question to SVO tuple

Most questions translate well to SVO tuples, while retaining important details about the question. These SVO tuples for each individual question can then be combined to create a knowledge graph. **Fig 7** shows a sample article and questions, with their respective SVO graphs. This method works quite well, as questions tend to be straightforward, and interrogative words such as "who", "what", "where" etc can simply be replaced by the given answer.

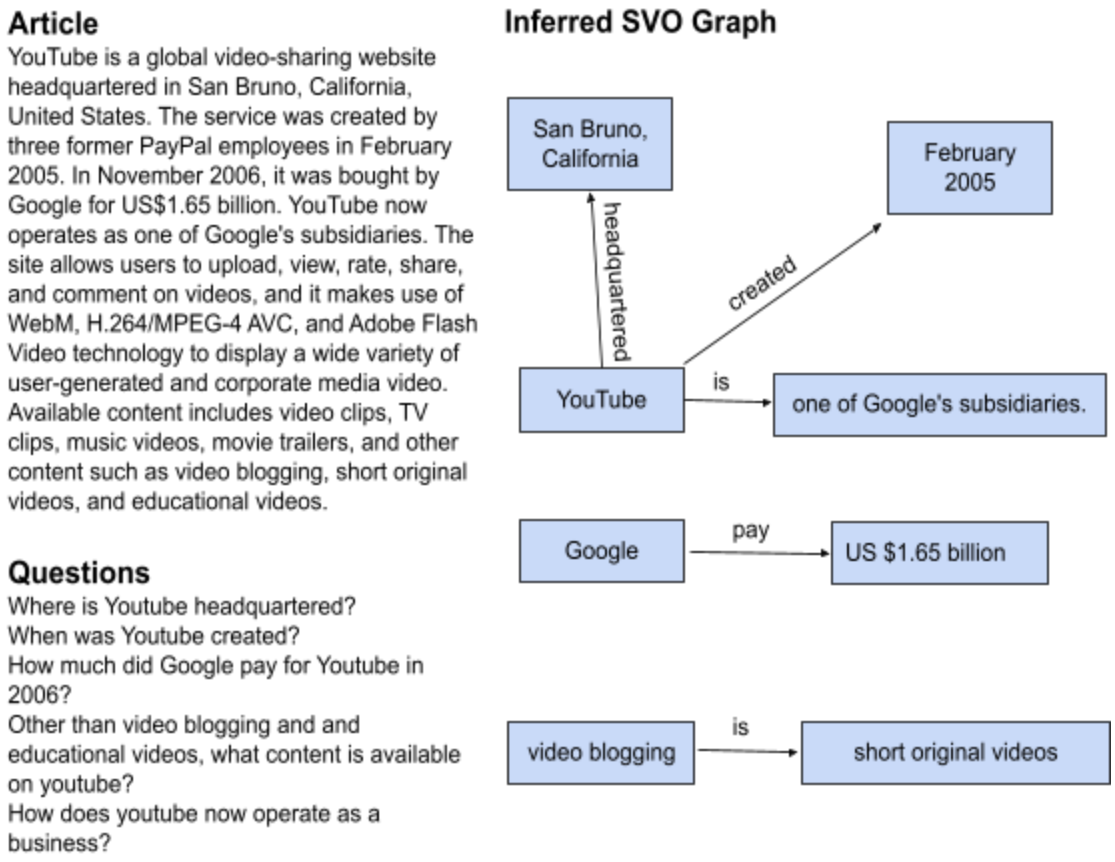


**Fig 7.** Example SQuAD Article and inferred SVO graph

### Network Architecture

Now that we have the ability to generate a dataset of inferred SVO tuples for a given article, we can train a new transformer network to generate this SVO knowledge graph from a passage of text. The system can be trained by leveraging a pre-trained NLP model such as GPT-2 **[17]** and applying transfer learning techniques to generate subject, verb, object knowledge graph tuples.

### Implementation of Experiment & Parameters

The fine-tuned NLP model will be trained to output SVO graphs for a given article. The SVO graph will be translated into a textual format in order to simplify the fine-tuning of the model. A pre-trained model will be utilized in order to increase accuracy and reduce training time.

### Fine Tuning & Transfer Learning

The T5 NLP model has been trained to be optimized for transfer learning tasks, such as text translation, summarization and generation **[19]**. The T5 model was trained using a transformer architecture using unsupervised learning methods on the Common Crawl dataset that contains over 2.8 billion webpages **[19]**. The model's architecture is capable of performing most sequence to sequence text tasks. This model is ideal for fine tuning, as it was trained on a variety of sequence to sequence tasks and was able to achieve state of the art results on many language tasks **[19]**.

The T5 model is pre-trained for text to text tasks, which can be leveraged for our model. Using transfer learning will reduce the training time and improve the accuracy of the model. We can use the same method used for text summarization fine-tuning in order to generate the SVO graph. Text summarization tasks are optimized to read in a piece of text as input, and to

output a much shorter piece of text that is representative of the contents of the passage. The pre-trained model will be fine-tuned to accept an article of text as input, and output a summarized SVO graph in a textual format.

### SVO Conversion Format

The pre-trained model needs a passage of text as input, and an annotated passage of text for training. The SVO dataset we have created has a SQuAD article of text as the input, but utilizes an SVO graph as the training output samples. It would be difficult to create an architecture that outputs graphs, since the baseline architecture utilizes a sequence to sequence transformer. However, SVO graphs can be converted to text using a similar technique to how the SVO dataset was generated. This method is demonstrated in **Fig 8**, and allows the fine-tuned model to generate a SVO graph from a piece of text. Each SVO tuple can simply be concatenated into a simple sentence structure as demonstrated below:

(“youtube”,”created”,”February 2005”) → “Youtube created February 2005.”

This format will create a short paragraph that can be used by the pre-trained model in order to optimize for the task of generating SVO graphs. In order to

parse the output text back to SVO, we simply use the spaCy library to parse the sentences into SVO tuples again.

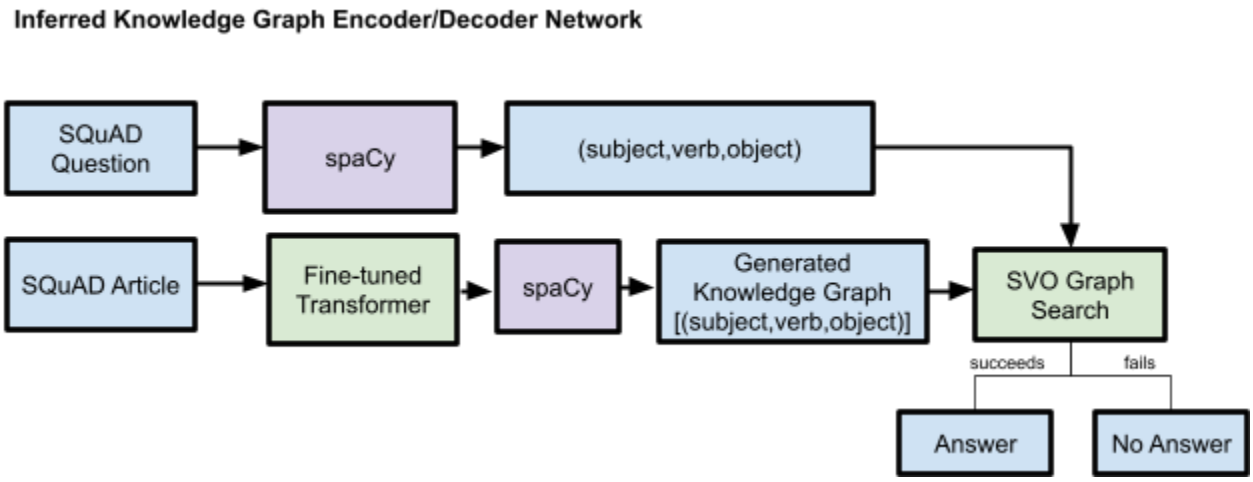


**Fig 8.** Architecture of experiment, with SVO parsing and inferred knowledge graph generation

### Baseline Comparison

In order to test the efficacy of our model, we will need a baseline model to compare it to. Many of the models from the SQuAD Leaderboard are trained using thousands of hours of compute time **[8]**. This is infeasible for our project, so we need to create a baseline that has the same architecture and training time. We can compare our model to a model trained to directly output answers on the SQuAD dataset. Using the same base model for fine-tuning will guarantee that any analysis of the performance of the SVO model will relate to the intermediate

representation, and not depend on outlying factors such as training time and model size.

### Evaluation

The network will generate a SVO knowledge graph for a given article. Then, each question will be parsed into a (subject, verb, object) tuple using the spaCy toolkit. We will run a search on the SVO graph to attempt to fill in either the subject or object. We will use both F1 and exact match scores to determine how successful the network is at generating SVO tuples. This will be compared to the T5 baseline model fine-tuned for SQuAD question answering **[18]**.

Another benefit of this methodology is that it doesn't generate knowledge that can't be inferred. There are some questions in the SQuAD dataset that have no answer that can be inferred from the article. Since our system will be unable to generate these, it will also fail to find an answer. This is more intuitive, as previous Q&A systems have to learn to determine when there is no given answer, while our system simply will not have any reference to it in the knowledge graph.

Models with intermediate representations are by their very nature difficult to train. Currently, the best performing model on the SQuAD leader board has achieved a F1 accuracy of over 93% **[8]**. An intermediate representation is more difficult to

train as it has to generate a representation that is not directly correlated with the exact answer. As previously noted, this creates a tradeoff between the model's accuracy and its ability to generalize. The T5 model fine tuned on SQuAD was able to achieve an F1 accuracy of ~81%. We expect the SVO model to receive an F1 accuracy of 70%, due to having an intermediate representation and generating answers without the explicit context of the questions. These results will reinforce the use of interpretable models in NLP tasks.

## Results

The SVO model and baseline T5-base model were trained on 130319 instances and evaluated on 11873 questions. The exact match score is calculated as the percentage of answers that were generated exactly as annotated in the dataset. The F1 score measures the precision and recall of the model by measuring the errors in token predictions. The results for both models tested on the validation dataset are shown in **Fig 9**.

| **Model** | **F1** | **Exact Match (EM)** |
|---|---|---|
| t5-base SQuAD (baseline) **[18]** | 81.32% | 77.64% |
| Fine-tuned SVO Network | 62.47% | 58.28% |

**Fig 9.** Model results for text generation from baseline and SVO models

The fine-tuned SVO model was not able to achieve the hypothesized accuracy of 70%. As expected, the SVO network performed slightly worse than the baseline model, due to its use of a learned representation.

### Sources of Error

The main sources of error for the SVO model comes from grammatical issues, not having the context of questions and model size. Many of these issues could be resolved through the use of a manually curated dataset or changes to the network architecture.

#### Grammatical Structure

One potential source of error in the SVO model relates to the grammatical structure of the dataset. Since the SVO tuples do not represent correct grammatical structure, the pre-trained model may struggle to generate the graph format. For example, the converted SVO tuple “Youtube created February 2005.” should actually be represented as “Youtube was created in February 2005.” in order to be correct in English grammar. Since the base model was trained on correct syntax, it will have difficulty generating these SVO tuples that are grammatically incorrect.

### Dataset Generation

The method used to generate the SVO dataset caused errors in the model. Since the SVO tuples were generated programmatically, some annotated knowledge graphs contained errors or were too abstract to be useful. Some SVO graphs contained duplicate (subject, verb) tuples, which prevented matching on the object when retrieving answers. For example, the SVO network generated the following tuples which both matched for a given question:

(“Los Angeles”,”is”,“the most populous city in California”)

(“Los Angeles”,”is”,”the second most populous city in the United States”)

In other cases, pronouns prevented the SVO model from correctly identifying an answer, as the graph did not contain an explicit reference to the subject, such as:

(“she”,”is”,”lead singer”) instead of (“'Beyoncé'”,”is”,”lead singer”)

This pronoun issue could likely be resolved through the use of a coreference resolution algorithm. The errors in the dataset contributed to errors in the model’s predictions.

### Context Sensitive Information

The context sensitive nature of the SQuAD questions also contributed to the errors in the SVO model. Models that perform well on SQuAD have access to the questions that will be asked when generating the answers **[2]**. Our SVO model

did not have any contextual information about the questions that would be asked when it generates the SVO graph. This led the model to generate some correct SVO tuples that had no utility in the question answering task.

An interesting example of this unintentional knowledge inference is shown in **Fig 10**. The SVO model only received an EM accuracy of 58% as it failed to generate some necessary tuples. Despite this, the other tuple in the SVO graph is a fact taken from the article. As shown in the highlighted section of the passage, the tuple (“seismic discontinuities”,”is at”,”410 and 610 kilometers”) is a valid tuple extracted from the article, even though it does not aid in the question answering task.

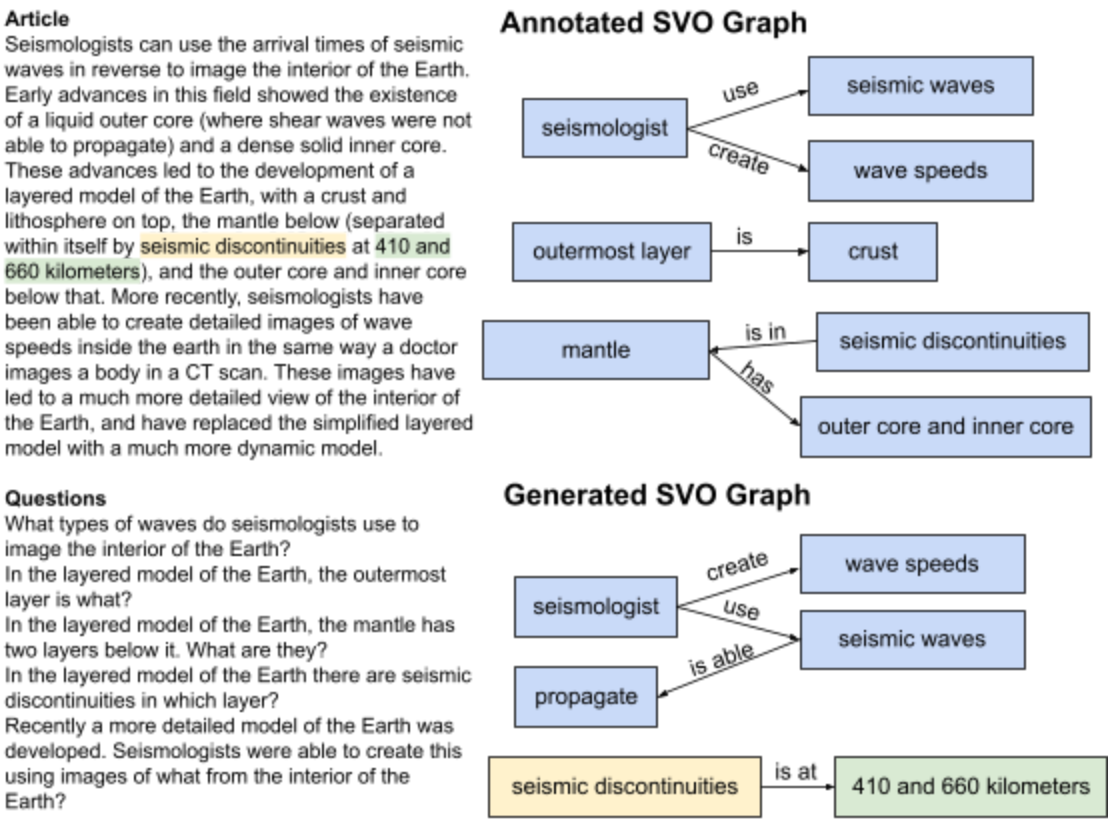


**Fig 10.** Example SQuAD Article, annotated SVO graph and SVO graph generated from the fine-tuned model. The tuple generated in the SVO graph that was not part of the annotated dataset is still factually correct.

## Implications of research

The SVO network was able to create interpretable SVO graphs, without relying on context sensitive questions or black box representations. This SVO model has proven that it is possible to create neural models for question answering that use an interpretable intermediate representation. Despite performing worse than the baseline model, the SVO model was still fairly accurate and able to answer SQuAD questions. The SVO network could likely replace black box models with some improvements to the dataset and network architecture.

## Future Work

The evaluation of the SVO model for question answering has exposed many new potential methods to explore for intermediate representations. The model size can be increased in order to improve performance and accuracy. The dataset can also be advanced, which can improve the fine-tuning of the model and its expressiveness.

Models generally perform better as the size and training time increases **[17]**. One clear area of improvement for the fine-tuned SVO model is to train it on a larger model, such as the T5-large or possibly even a GPT-2 model. The T5-11b model was able to achieve an F1 score of over 96% when fine tuned on the SQuAD dataset **[19]**. This is an increase of ~15% from the T5-base model that was used

as the baseline in this experiment. If we fine-tune our SVO model using the T5-11b model, it is likely we will realize similar gains in performance. This possible higher accuracy would place our SVO model closer to state of the art question answering systems, at the cost of more training time and computational resources.

The dataset used for fine-tuning the SVO model could be improved to better aid in text generation. A manually annotated dataset that includes many facts that can be inferred from a SQuAD article would increase the efficacy of the model, and ensure that it can generate the answers to all questions without knowing the context beforehand. Annotators could create simple factual SVO statements from a given article in order to aid the fine-tuning and ensure that the most information possible can be extracted from a model.

The model could also be improved through the use of context in the SVO graph generation. As noted in **Fig 10,** the model generated SVO tuples that were correct but not relevant to the question answering task. The model can be better constrained to the given task by including the questions that will be posed into the input of the SVO model. This would allow the model to better generate SVO tuples that are relevant to the question that will be asked. However, this might reduce the generalization of the model, as it won't be as likely to generate facts

that are not necessary to answer the posed questions. Including the questions also requires all of the questions generated to be known before evaluation, as the model will not extrapolate to features outside of the given constraints.

## Conclusion

In this project, we explored the use of interpretable intermediate representations for question answering tasks. Through the use of a fine-tuned neural network, we were able to create a system that can generate SVO graphs from a passage of text that can be used to answer questions.  This system is interpretable and allows a human expert to curate or edit the dataset to further remove errors. The analysis of the experiment demonstrated the inherent difficulty in training an interpretable model. Despite not reaching state of the art performance, the system was able to accurately generate SVO graphs and could likely improve through tweaks to the dataset and network architecture. We hope to promote future research in interpretable question answering models by releasing the dataset and models associated with this paper.

## References

**[1]** Singh, H., Aggrawal, M., & Krishnamurthy, B. (2020). Exploring Neural Models for Parsing Natural Language into First-Order Logic. arXiv preprint arXiv:2002.06544.

**[2]** Rajpurkar, P., Jia, R., & Liang, P. (2018). Know what you don't know: Unanswerable questions for SQuAD. arXiv preprint arXiv:1806.03822.

**[3]** Gardner, M., Talukdar, P., Kisiel, B., & Mitchell, T. (2013, October). Improving learning and inference in a large knowledge-base using latent syntactic cues. In Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (pp. 833-838).

**[4]** Wang, P. (2013, July). Natural language processing by reasoning and learning. In International Conference on Artificial General Intelligence (pp. 160-169). Springer, Berlin, Heidelberg.

**[5]** Zelle, J. M., & Mooney, R. J. (1996, August). Learning to parse database queries using inductive logic programming. In Proceedings of the national conference on artificial intelligence (pp. 1050-1055).

**[6]** Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.

**[7]** Dali, L., & Fortuna, B. (2008). Triplet extraction from sentences using svm. Proceedings of SiKDD, 2008.

**[8]** Rajpurkar, P. (n.d.). The Stanford Question Answering Dataset. SQuAD. https://rajpurkar.github.io/SQuAD-explorer/

**[9]** spaCy · Industrial-strength Natural Language Processing in Python. (n.d.). SpaCy. https://spacy.io/

**[10]** DBpedia. (n.d.). DBpedia. https://wiki.dbpedia.org/

**[11]** Zettlemoyer, L. S., & Collins, M. (2012). Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. arXiv preprint arXiv:1207.1420.

**[12]** Wikidata. (n.d.). Wikidata. https://www.wikidata.org/wiki/Wikidata:Main_Page

**[13]** Luger, G. F. (1994). By George F. Luger - Cognitive Science: The Science of Intelligent Systems (1994-06-23) [Hardcover]. Academic Press.

**[14]** Luger, G. F., & Stubblefield, W. A. (1990). Artificial intelligence and the design of expert systems. Benjamin-Cummings Publishing Co., Inc.

**[15]** Sultana, T., & Badugu, S. (2020). A review on different question answering system approaches. In Advances in Decision Sciences, Image Processing, Security and Computer Vision (pp. 579-586). Springer, Cham.

**[16]** Zhang, Z., Yang, J., & Zhao, H. (2020). Retrospective reader for machine reading comprehension. arXiv preprint arXiv:2001.09694.

**[17]** Radford, A., Wu, J., Amodei, D., Amodei, D., Clark, J., Brundage, M., & Sutskever, I. (2019). Better language models and their implications. OpenAI Blog https://openai. com/blog/better-language-models.

**[18]** T5-Base Fine-Tuned Results. (n.d.). Retrieved March 17, 2021, from https://huggingface.co/mrm8488/t5-base-finetuned-squadv2

**[19]** Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., ... & Liu, P. J. (2019). Exploring the limits of transfer learning with a unified text-to-text transformer. arXiv preprint arXiv:1910.10683.

**[20]** Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. arXiv preprint arXiv:1706.03762.

**[21]** Squad2.0. (n.d.). Retrieved March 24, 2021, from https://rajpurkar.github.io/SQuAD-explorer/explore/v2.0/dev/Normans.html?model=nlnet+%28single+model%29+%28Microsoft+Research+Asia%29&version=v2.0