Dictionary-Based Data Generation for Fine-Tuning BERT for Paraphrasing Tasks

Dr. Istvan Lauko and Mark Anthony Carthon III

Spring 2020

1 Chapter 0: History of Natural Language Processing (NLP)

The field of NLP comes from a field called Machine Translation (MT). Machine Translation is the study of how to algorithmically translate one language to another. One of the first people to work in Machine Translation a Persian intellectual named Al-Kindi (circa 801 AD - 873 AD). Al-Kindi was born in a city named Kufa, but was educated in Bagdad. He was a prominent figure in the Grand Library of Bagdad. This was a public academy and the intellectual center of Bagdad during the Islamic Golden Age. A number of Abassid caliphs appointed Al-Kindi to oversee the translation of Greek scientific and philosophical texts into the Arabic language. Al-Kindi developed various methods to do this. These methods relied on the frequency of the characters that were used in the texts. He developed techniques for systematic langauge translation including, but not limited to: crytoanalysis, frequency analysis, probability theory, and statistics, all of which are used in modern machine translation.

Not much happened in MT after Al-Kindi until 1623 when Rene Descartes (and others) proposed an artificial universal language; in this language, equivalent ideas in different languages would share the same symbol. Both Leibniz and Descartes put forward proposals for codes which would relate words between languages. Of course, all of these proposals were purely theoretical, and none of them resulted in the development of an actual algorithm for language translation.

Then not much happened in MT until the mid-1930s when thee first patents for "translating machines" were applied for. One proposal by Georges Antsrouni was simply an automatic bilingual dictionary using papertape. Papertape was a form of data storage that consisted of a long strip of paper in which holes were punched. Papertape was used mostly during the 19th and 20th centuries. They were effective with teleprinter communicationas input for computers of the 1950s ad 1960s, and later as a storage medium for minicomputers and CNC machine tools. Another proposal was made by Peter Troyanski, a Russian who included a bilingual dictionary, and a method for dealing with grammatical roles between languages based on Esperanto.

Esperanto was the most widely spoken constructed international auxiliary language. It was created by a Polish opthamologist called Ludwik Lejzer Zamenhof in 1887. The purpose of this universal language Esperanto was to create a flexible langauge that would serve as a universal second language to foster world peace, international understanding, and to build a "community of speakers".

Peter Troyanski's proposal was a system that was separated into three stages. The first stage consisted of a native-speaking editor in the source language to organize the words into their logical forms and to exercise the syntactic functions. A logical form of a syntactic expression is a precisely-specified semantic version of that expression in some formal system. Think of this first stage as tokenization for BERT. The second stage of Troyanski's proposal was for the machine (such a machine didn't exist when this proposal was initially made) to translate these logical forms into the target language. Stage three required a native-speaking editor to normalize this output. This system didn't become relevant on a big scale until the 1950s when computers became widely utilized.

In 1946, Andrew Donald Booth, a British electrical engineer, physicist, and computer scientist, proposed the idea of using digital computers for translation of natural languages. Warren Weaver was a researcher at the Rockefeller Foundation and he presented one of the first set of computer based machine translation proposals in 1949; it was called "Translation Memorandum". Warren Weaver proposals came about thanks to advancements in information theory, successes in cryptography during World War II, and theories about invariants at the foundation of natural language. In 1950, a major breakthrough was made by Alan Turing when he published his famous paper "Computing Machinery and Intelligence" which proposed the "Turing Test" as a criterion of intelligence. This criterion depends on the ability of a computer program to impersonate a human in a real-time written conversation with a human judge, sufficiently well that the judge is unable to reliably distinguish between the computer program and a real human on the basis of conversational content alone.

In 1951, Yehosha Bar-Hillel began research in MT and later claimed that without a "universal encyclopedia" a machine would never be able to deal with the problem of one word having multiple definitions because during this time, semantic ambiguity could only be solved by writing source texts for machine translation in a controlled language that uses a vocabulary in which each word has exactly one meaning. Also in 1954, Georgetown University who created an MT research team. On January 7, 1954 Georgetown and IBM teamed up to give a public demonstration of its Georgetown-IBM experiment system which involved fully automatic translation of more than 60 Russian sentences into English. This demonstration was widely reported in the newspapers and

gained public interest. However, this feat was not all it was made out to be because it had only 250 words and translated 49 carefully selected Russian sentences into English. Most of the words were related to the field of Chemistry. Despite its shortcomings, this demonstration encouraged others to put more resources into furthering MT on a worldwide scale.

Andrew Donald Booth designed the All Purpose Electronic (X) Computer, APE(X)C, at Birkbeck College (now the University of London). In 1954 a demonstration was made on the APE(X)C of a rudimentary translation of English into French. In 1955, Japan and Russia began to develop their own MT research programs and in 1956 the first ever MT conference was held in London. In 1957, Noam Chomsky, an American linguist developed the idea of syntactic structures which was a work that sought to prove that semantics and syntax are two separate concepts. He did this by forming the sentence "Colorless green ideas sleep furiously", which shows that semantics and syntax really are different.

In the 1960s, some NLP processing systems were developed, such as: SHDRLU which was a natural language system that worked in restricted "block worlds" with restricted vocabularies. This block world was a breakthrough in Artificial Intelligence (AI) which made it easier for systems to model and reason about abstract symbols. Also, ELIZA was another NLP processing system which was a simulation of a psychotherapist. This was written by Joseph Weizenbaum who was a German-American computer scientist from MIT. ELIZA was able to provide human-like interaction whenever the patient was saying things within ELIZA's small database. This was quite remarkable considering that ELIZA was given no information about human thought or emotion.

During the 1960s, The United States and the Soviet Union did research in Russian - English translation of scientific journals and technical articles. This endeavor was successful in that the rough translations helped to give a basic understanding of the material. In 1962, the Association for Machine Translation and Computational Linguistics was formed in the U.S.A. and many researchers joined it. Research in MT continued until the US Government commissioned the Automatic Language Processing Advisory Committee (ALPAC) report. This committee was made up of seven scientists who decided that there was a lack of progress in the last ten years of MT research despite significant funds being allocated toward this research area. This ALPAC report concluded that MT was more expensive, less accurate, and slower than a human translator. This report also concluded that MT was unlikely to make very much progress towards human-level translation. Consequently, MT funding was greatly reduced. There was a beacon of hope though because this ALPAC report recommended that tools (i.e. automatic dictionaries) be developed to aid human translators and that some research in computational linguistics should

continue to receive funding and support.

Because of this report, MT research was practically abandoned for over a decade. This had a ripple effect as the Soviet Union and the United Kingdom because to cut back on their MT research efforts as well. However, Canada, France, and Germany continued to research in MT. In 1969, Roger Schank, an American A.I. theorist, introduced the conceptual dependecy theory for NLU; this model was partially inspired by the work of Sydney Lamb, an American linguist. The biggest efforts in MT in the US, during the period after the ALPAC, was given by Systran and Logos; these companies did MT work for the US Department of Defense. In 1970, the Systran system was installed for the Us Air Force, it was used by the Commission of the European Communities in 1976, and Xerox used Systran to translate technical manuals in 1978. Also in 1970, William A. Woods developed the augmented transition network (ATN) to represent natural language input. These ATNs were graph-theoretic structures that are used in formal languages. Instead of phrase structure rules, ATNs used an equivalent set of recursively called finite state automata. Eventually, these ATNs were generalized into so-called 'generalized ATNs". In 1970, The French Textile Institute used MT to translate abstracts between French, English, German, and Spanish. In 1971, Brigham Young University (BYU) started a project to translate Mormon texts automatically.

During the 1970s, it was common for computer programmers to write 'conceptual ontologies', which structured real-world information into data that was of such a form that a computer would be able to understand what it was. Some examples of this is: MARGIE (Schank, 1975), SAM (Cullingford, 1978), and many more. In the 1960s, MT research concentrated on limited language pairs and input, but in the 1970s, the concentration was on low-cost systems that could translate technical and commercial documents. Up to the 1908s, most NLP systems were based on complex sets of handwritten rules. By the 1980s, both the diversity and the number of installed systems for machine translation had increased. A number of systems that relied on mainframe technology were in use (i.e. Systran and Logos). Systran's first implementation system was implemented in 1988 by the French Postal Service's online service called Mintel. There was a market for lower-end machine translation systems because of the much improved availability of microcomputers. Thus, microcomputers became more common in Europe, Japan, USA, China, Korea, and the Soviet Union. Japan had a fifth generation computer and tried to create software for translating between English and many other Asian languages. During the 1980s, translation relied on intermediary linguistic representation such as: syntactic, morphological, and semantic analysis.

In the late 1980s, computational power increased because of Moore's Law and the fact that researchers were relying on Chomskyan theories less because it's theory discouraged the kind of

corpus linguistics that is at the foundation of the machine learning approach to language processing. As a result, computational power also became less expensive. Also, more research and attention was allocated towards statistical models for machine translation. Some of the earliest used machine learning algorithms (i.e. decision trees) produced systems difficult if-then rules which were similar to existing hand-written rules, but the statistical models made soft, probabilistic decisions based on attaching real-valued weights to the features of the input data. In the 1990s, MT began to move away from large mainframe computers and towards personal computers and workstations. Recent research has focused more on unsupervised and semi-supervised learning algorithms. These algorithms learn from data that has not been annotated or a combination of non-annotated and annotated data with the desired answers given and with the creation of the World Wide Web, lots of non-annotated data is available.

In the 2010s, representation learning and deep neural network machine learning methods because widespread in NLP. This is because these methods have been shown to produce state-of-the-art results in many NLP tasks such as: language modeling, parsing, etc. Many popular techniques use word embeddings to capture semantic properties of words and an increase in end-to-end learning of higher level tasks.

2 Applications of NLP

Natural Language Processing (NLP) is a sub-topic of Artificial Intelligence and it's purpose is to train computers to understand, interpret, read, hear, and communicate using human (natural) languages. In 1997, Long Short-Term Memory (LSTM) Recurrent Neural Network (RNN) models were introduced and found great success with voice and text recognition, and text and speech generation. RNN models use previous outputs as inputs for furture iterations of the algorithm. In 2001, Yoshio Bengio and his team proposed the first neural language model using a feed-forward neural network (FFNN). The big difference between RNNs and FFNNs are that RNNs use previous outputs as future input, but FFNNs do not, the data is feed forward, an output is produced, then the new input is fed forward and so on.

In 2011, Apple released Siri which became known as the world's first successful NLP assistant to be used for the general public. The Automated Speech Recognition module in Siri can translate the user's words into digially interpreted tokens/concepts. Siri has predefined commands within its software and uses the tokens it receives from the user's speech and matches it up with one or more of its predefined commands. Then it carries out the said commands. Machine Learning techniques are necessary because NLP engines need to recognize a command even if the command is not expressed using the exact sequence of tokens that the engine has been

programmed with; it has to allow for some freedom of expression while still capturing the main idea of the user's words and matching it to the predefined set of commands.

In November 2014, Amazon announced Alexa alongside Echo. The inspiration for Alexa stems from the computer voice and conversational system on the Starship Enterprise in Sci-fi works, startig with Start Trek. One of the reasons the Amazon developers chose the name Alexa is because x is a hard consonant and hard consonants are recognized with higher precision than other types of letters. The Amazon developers also liked the name Alexa because it was reminiscent of the Library of Alexandria, which is where the field of MT began (the precursor to NLP).

Google Assistant was unveiled during Google's developer conference on May 18, 2016. It was part of the unveiling of the Google Home smart speaker and a new messaging app called Allo. Google's CEO claimed that Google Assistant was designed to be a conversational experience, a two-way experience, and "an ambient experience that extends across devices".

3 Chapter 1: Introduction

IGL– Define NLP, NLU, discuss its importance its applicability.
IGL– Discuss briefly the history of natural language processing of and in more detail the recent advances and the current state of the art in NLP (variants of transformer networks).
IGL– Introduce the network BERT discuss its structure and capabilities (use basic references).
IGL – QQP and MRPC are paraphrase datasets that consist of sentence pairs.

The goal of this thesis is to investigate the facilitation of the training of large neural networks to understand (i.e.: develop the capacity to identify and differentiate) grammatical structure and extended vocabulary, including a wide range of idiomatic language use in the English language. More specifically, we would aim to re-train a Bidirectional Encoder Representations from Transformers (BERT) neural network to enhance its capacity to understand the language well enough to be able to paraphrase a large variety of sentences. The task of computational paraphrase identification, a high-level artificial intelligence task in the language understanding domain (NLU), is to predict whether the two sentences in a sentence pair are semantically equivalent.

In order to either train from scratch or to retrain/fine tune for close to human performance on a complex task such large pre-trained networks in a supervised training regime, extensive amount of labeled training data is needed. For strong performance of the targetted trained network in a

general NLP setting the training data deployed need to represent a rich variety of the targeted language use. Training such large systems is expected to require millions of annotated training samples that show a representative distribution of the targeted language use. Such richness in the training data would be key to improve the performance of state of the art systems in automated language processing and language understanding, but currently for most tasks it is not available in the necessary volume and quality.

Development of the needed training data sets with the current methods (i.e.: based on hand-labeling by humans) are prohibitively expensive: it would require hundreds of thousands of expert human work-hours or possibly substantially more. Thus current industry standard data sets for training and evaluation for a number of NLP tasks are relatively small. For instance the General Language Understanding Evaluation (GLUE) benchmark's MRPC component to evaluate a network's capacity to train/perform for the task of paraphrasing is relatively small (on the order of 5000 sentences) and it is also restricted to limited a (journalistic) language domain.

One of the objectives in this thesis is to explore a method to generate annotated training data in large volume to help to train ot fine tune trained networks in supervised learning setting.

We base our paraphrase sample generation method on content extracted from an extensive synonym dictionary: The Oxford Thesaurus, An A-Z Dictionary of Synonyms, Oxford University Press (1994) (OTAZ). This dictionary puts a focus on compiling complete synonymic word and expression sets with an emphasis on representing rarer word and expressions uses, associated with semantic content of synonym sets. That information is very hard for networks to learn from corpus (even from billion-word corpora) due to very low frequency use even in very large text collections (corpus). However, the most commonly known synonyms may have been omitted from the collection to limit the volume of the original publication. Another attractive and more complete option as a source for the adverbial, adjective, noun and verbal synonym set could be WordNet database, however WordNet does not contain rarer part of speech sets (such as Interjections, Conjunctions,...) of words and expressions and is more limited in the number of example sentences listed to illustrate semantic function.

Additionally to its polysemous synonymic organization of its headwords, OTAZ provides additional stylistic and usage information on the synonym sets and importantly a rich set of example sentences (1-4 per synonym sets) using the headword of each synonym set entry.

The idea of sentence pair generation using lexicographical resources for enhanced training of neural language networks for classification of sentence pairs relies on substitution of synonymic word and expression into example sentences from thesauruses, synonym dictionaries and similar

lexicographical resources. In OT Fundamentally such a substitution task poses challenges of varying complexities depending on the part of speech categories the synonym sets belong to. The difficulties are ranging from the simple, (e.g.: for the case of adverbial synonym sets with very little to no mophological variation induced with substitution, to the sophisticated in the case of synonymic verbal expressions that are required to be adjusted in number, tense and attracted noun cases along during a substitution step to provide correct sentence pairs.

In this work, for simplicity, we limit our investigation for generation and training use of adverbial synonym sets only.

There was not a substantial amount of data online for download, so we had to create our own training data. We used text data from an Oxford synonym dictionary. The format of this dictionary was that there was a given word, its part-of-speech, some synonyms, and an example sentence. Dr. Lauko scanned the text from the dictionary to create a text file that had all of the contents of the dictionary. Then we used regular expressions to organize the text. Using regular expressions, we eliminated undesirable characters in the text and structured it so that we had one column for the synonyms and another column for the example sentences. The example sentences contained the first synonym in each row. You can think of this as a $n \times 2$ matrix. This text document was loaded into a BERT network for training.

4 Chapter 2: Regular Expressions

A regular expression (REGEXP) is a sequence of characters that define a search pattern. Usually such patterns are used by string searching algorithms for "find" or "find and replace" operations on strings, or for input validation. It originated in the 1950s and mathematician [from the U.S.A.] Stephen Cole Keene is given credit for creating it. Here are some basic character strings that are used in reglar expressions:

$\backslash d$	searches for one digit 0 through 9
$\backslash w$	searches for any word character A-Z and a-z
$\backslash s$	searches for any whitespace character
$+$	searches for a given regexp one or more times
$*$	searches for a given regexp zero or more times
?	searches for a given regexp only one time or no times

For example, if you have a text document and you want to search for a word character followed by a digit one or more times, you would use the regexp: $\backslash w\backslash d+$.

We used regular expressions to "clean up" the text documents that we wanted to used for training. We deleted all unnecessary and undesirable characters in the text, we rearranged words in the text, and we created multiple example sentences from the given example sentence using regular expressions. REGEXP made this task easier because it allowed us to make edits on very large text documents without having to manually go through single line. We were able to check that everything worked in the desired way by changing a few lines at a time, first, then once we were confident we would change the entire document. Of course, it is imperative that one is very familiar with the document before using regular expressions because you want to be able to know what is in the document so that you know exactly what needs to be done and how complicated it would be.
A snippet of the reformulated lexicographical database obtained from OT is represented below. The database in stored in a text file is line based with unicode character sets. Here

```
Ð, @, &, %, #
```

and ~ represent specially selected, unprintable separator Unicode characters not appearing in the text-based data of our the database which are used as special markers. Each paraghraph in the representation below, headed by a lead expression and ending with one or more example sentences, shows an adverbial synonym set and its use. To a single lead expression several there me be associated several alternative use sense or synonym set.
Each paragraph below is in fact a line in the database ending in a newline character. This representation allows us to use regular expressions for complex data generation very efficiently.

```
Ðabout+@adv.&4* %about, here_and_there, far_and_wide, hither_and_yon,
hither_and_thither, helter-skelter,
: My papers were scattered about# as if a tornado had struck.~

Ðabout+@adv.&5* %about, around, prevalent, in_the_air,
: There is a lot of flu about# this year.~

Ðabout+@adv.&6* %about, approximately, nearly, close_to, not_far_from,
almost, just_about, around, : It is about# time you telephoned your
mother.~

Ðabove+@adv.&1* %above, overhead, on_high, aloft, in_the_sky,
in_the_heavens,
: Far above#, the clouds scudded swiftly by.~
```

Đabove+@adv.&2* %above, upstairs,
: They lived on the ground floor and the landlady lived above#.˜

Đabove-board+@adv.&1* %above-board, openly, candidly, freely, publicly, frankly, straightforwardly, plainly, for_all_to_see, out_in_the_open, in_the_open,
: Donald has always dealt completely above-board# with everyone.˜

Đabroad+@adv.&1* %abroad, overseas, in_foreign_lands, in_foreign_parts,
: We were abroad# on assignment for a few years.˜

Đabroad+@adv.&2* %abroad, broadly, widely, at_large, near_and_far, far_and_wide, everywhere, extensively, publicly,
: Don’t spread rumours abroad#.˜ : Don’t spread rumours abroad#.˜

Đabroad+@adv.&3* %abroad, outside, out_of_doors, away, out_and_about,
: There are few people abroad# this early in the morning.˜

Đabsolutely+@adv.&1* %absolutely, unqualifiedly, unconditionally, unreservedly, unexceptionally, unequivocally, unquestionably, positively, definitely, really, genuinely, decidedly, surely, truly, certainly, categorically,: She is absolutely# the best dancer I have ever seen.˜

Đabsolutely+@adv.&1* %absolutely, unqualifiedly, unconditionally, unreservedly, unexceptionally, unequivocally, unquestionably, positively, definitely, really, genuinely, decidedly, surely, truly, certainly, categorically, :: I absolutely# refuse to go.˜

Đabsolutely+@adv.&2* %absolutely, totally, utterly, completely, entirely, fully, quite, altogether, wholly, : It is absolutely# necessary that you undergo surgery.˜

Đaccordingly+@adv.&1* %accordingly, hence, therefore, consequently,

thus,in_consequence_of, in_consequence_whereof, so, and_so, : Smoking was forbidden; accordingly#, we put out our cigars.˜

Ðaccordingly+@adv.&2* %accordingly, suitably, in_conformity, in_compliance, conformably, appropriately, compliantly, : Dinner-jackets were required, and the men dressed accordingly#.˜

Ðactually+@adv.&1* %actually, really, in_reality, in_fact, in_actuality, in_point_of_fact, in_truth, absolutely, as_a_matter_of_fact, indeed, truly, literally, : The interest rates actually# charged by banks may vary from those quoted publicly.˜

5 Chapter 3: Structure of Transformer Networks

The BERT network is a type of Transformer network. A Transformer network is a neural network that has both encoders and decoders (BERT only has encoders) and was introduced in the well-known paper "Attention is All You Need". Transformer networks were used for sentence translation. Your input would be a French sentence, for example, and the Transformer would translate the French sentence into English, and then output the English translation. The architecture of the Transformers is that it has a stack of encoders and a stack of decoders. The encoders basically take the French sentence and transform it into the language of the network , and then the last encoder feeds this information into the stack of decoders. The decoders take the information from the last encoder and they turn the information from the language the network understands into the the desired language of the user (which is English in this example).

These encoders can turn the input into language the model understands via a technique called word embeddings. These word embeddings are the process that the network takes in order to take a given word and turn it into a vector of a certain size. When using the Transformer network, the word embeddings all have the same size. These vectors contain the word and some of its meaning. For example, the word 'King' and the word 'Queen' each have their own vector from the word embeddings. If you take the vector for King and subtract the vector for Queen, the difference will be a vector that is "small" in magnitude. These word embeddings are possible after a network has been trained. So, the encoders use word embeddings to gather the necessary information from the input. Each of the encoders has a structure to it.

In each encoder, there is a Self-Attention mechanism and a feed-forward network. The self-attention mechanism was invented in the paper "Attention is All You Need". The goal of

attention is to help the network to understand the connection that each of the words have with each other. For example, in the sentence: The dog took a bone a buried it in the ground. What does the word 'it' refer to in this sentence? Someone who is fluent in English would know that the word 'it' would refer to the bone. The goal of attention is to help the network to understand that the word it refers to the bone. This is why attention is so important; in order for the network to properly understand the sentences, it must understand how each word in a sentence relates to all of the other words.

How does the self-attention actually work? Self-attention requires a few things: word embeddings, queries, keys, and values. For each word embedding, a query, key, and value vector is created for it. Once a network has been trained, the network will have a weight matrix W_Q for the queries, a weight matrix for the keys W_K, and a weight matrix W_V for the values. Each word embedding vector is multiplied by each of the weight matrices. This would produce a query vector q_i, a key vector k_i, and a value vector v_i corresponding to the word embedding x_i. Next you take $q_1 \prod_i k_i$. Then you do $\frac{q_1 k_1}{+\sqrt{\dim(k)}}$ and take the "softmax". This softmax normalizes the constant so that it is between 0 and 1 (a probability). Then you take each value vector times the softmax score, then take the sum of all of them. This will give you a weighted sum of all of the value vectors. This calculation is very similar for matrices. Here it is:

$$X \times W_Q = Q$$
$$X \times W_K = K$$
$$X \times W_V = V$$
$$softmax\left(\frac{Q \times K^T}{+\sqrt{d_k}}\right) \times V = Z$$

In this calculation, X is the input, Q is the query, K is the key, V is the value, Z is the output/head.

There is also the notion of multi-headed attention. This notion helped the network's attention layer to improve its performance. With multi-headed attention, the network produces multiple attention heads or multiple Z matrices. In this process we have multiple $Q/V/K$ matrices. In the Transformer model, we have 8 attention heads. This will give us $Z_0, Z_1, ..., Z_7$. Since the model only expects one attention head, we concatenate all the $Z's$ together to get Z then multiply Z by another weight matrix W which was trained by the model to give us a Z of the desired dimensions. Why would multi-head attention be useful? Consider the sentence, "Dave bought a phone, but didn't know how to operate it." What is the word "it" referring to? Of course, we humans are fluent enough in language to know that the word "it" refers to the phone, but the model is not as fluent in language as we are. The multi-head attention aids the network to learn

that the word "it" refers to the phone.

As we all know, the order of words in a sentence is important. The network has a way of understanding the position of each token in a sentence by using a "positional encoding" that it learns through training. There are some residuals in the encoders. What happens is that the input X is fed into the encoder, self-attention takes place, the result of the self-attention Z is taken then the network computes $X + Z$, normalizes it, then feeds it into the feed-forward part of the encoder. The result of the feed-forward part of the encoder Z' is added to Z and normalized. Then this result Z'' is fed into the next encoder and the process repeats. Once our original input has made it's way through all of the encoders, it proceeds to the decoder.

The structure of the decoder is as follows: The output of the encoder is fed into the decoder. The decoder takes this through self-attention, adds and normalizes, then does an encoder-decoder attention, then adds and normalizes, then goes through the feed-forward, then adds and normalizes, then repeats this process as it gets pushed into the next decoder. Once the original output of the encoder layers gets pushed through all of the decoders, it goes through a linear layer, then gets its softmax score. This linear layer is what takes the vector that the decoders spit out and turns it into one of the tokens in its dictionary. It does this by projecting the output vector from the decoders into a logits vector whose length is the number of words in the given language that it knows. The softmax is used to help the network decide which of the words is the most probable.

6 Chapter 4: Project

The goal of this project was to train a BERT network to be able to recognize when a pair of sentences that contained adverbial words or expressions were paraphrases of each other.

```
The first task was to create data for a BERT Transformer. This data
needed to be example sentence pairs such that a sentence pair would
be labelled with a one if either sentence were a paraphrase of
the other. If one has a sentence pair such that neither sentence is a
paraphrase of the other, this pair is labelled with a zero.
In other words, the goal is to create sentence pairings as data to
train a neural network to learn to paraphrase. From where, and how,
can such data be obtained?

There is an Oxford synonym dictionary that contains words, synonyms
```

to those words, and example sentences using those words. An example of a line given in the dictionary is as follows: about adv.: about, here and there, far and wide, hither and yon, hither and thither, helter-skelter: My papers were scattered about as if a tornado had struck.

The presentation of every word in this synonym dictionary follows this basic structure. This looked like a promising source for data. As such, this entire dictionary was scanned and entered in a text editor for cleaning. Various special characters were inserted to make the cleaning process easier. For instance, the above example was changed to: Đabout+adv.4⋆ about, here_and_there, far_and_wide, hither_and_yon, hither_and_thither, helter-skelter, : My papers were scattered about as if a tornado had struck.

As you can see, the Đ marks the beginning of a new word, the + marks the end of the word, the marks the beginning of the part of speech, the marks the end of the part of speech, the number four denotes how many times this has been used (because many words have multiple meanings and each word that is in the dictionary has a separate synonym sequence for each different meaning it has), the ⋆ marks the end of the number, the marks the beginning of the first synonym. Multiword expressions have underscores to separate each word, although some terms are separated by hyphens, for example, helter-skelter. Each synonym is followed by a comma, even if it is the last synonym in the list. Each example sentence follows a colon and ends with a punctuation mark and a ta (musical note). A tab always separates the synonyms and the first example sentence. These characters make it easier to do regular expressions on the text because many of the symbols used are rarely seen in dictionaries and thus, are unlikely to clash with anything in the text originally. If ! was used instead of one of the special characters, there would have been a higher probability of error with the regular expressions because ! is a more commonly used character in dictionaries.

After the dictionary has been cleaned up and organised by the
special characters, sample sentences are created. For this, a regular
expression search pattern was created:
^([^\n\t]+)([^\n:\t,]+)([,\t]* *)([^\n:\t]+)(\t:+)([^\n:]*)([^\n:]++)([^\n]+)([^\n]+
along with the following regular expression replacement pattern: \1\2\3\4\5\6\7\8\9\5\

The following explains the search pattern:
^ means to start at the beginning of the line.
([^\n\t]+) means that the aim is to collect everything that is not an end of line, no

([^\n:\t,]+) means that a will be matched preceding the regexp in the parenthesis, whi

([,\t]* *) is denoted by \3 and collects everything until the next comma or tab, which

([^\n:\t]+) means \4 will collect everything after the second synonym that is not an e

(\t:+) means \5 will collect the tab and colon that occurs one or more times, then sto

([^\n:]*) means \6 will collect everything that is not an end of line, nor a colon if

([^\n:]++) means \7 will collect the keyword, which conveniently has a sigma at the en

([^\n]+) means \8 will collect everything after the \keyword", but stops before the ta

([^\n]*)\n indicates the search pattern that locates the ta , which marks the end of t

The replacement pattern is as follows: \1\2\3\4\5\6\7\8\9\5\6\2\8\n

This means that everything up to the first synonym is the same, but the gets moved to

It is important to note that marks the keyword in each example sentence after the ori

The first letter in any sentence should be capitalised. In order to assume this is the

So, (\t:+) means that \1 will collect a tab followed by a colon that occurs one or mo

The replacement pattern \1\U\2\E means that the one or more tabs remain in place in fr

At this point, the sentences have been created and rendered grammatically correct (exc

Step #1. Search pattern: , and the replacement pattern is: . The point of step one is

Step #2. Search pattern is as follows:

(\t:+)([^\n\t]+)([^\n]*)()(\t)([^\n\t]+)(\t)([^\n]*\n)

The component parts will now be addressed.

(\t:+) means that \1 will search for the , which is in between the last synonym and

([^\n\t]+) means that \2 will start after the colon in \1 and collect everything that

([^\n]*) means that \3 will collect everything in between the ta in \2 and the ti-ti

() as \4 is self-explanatory. (\t), which is \5 is also self-explanatory and is the ta

([^\n\t]+) means that \6 will search for and collect anything after the tab that is no

(\t) as \7 is the tab between the second and third example sentence.

([^\n]*\n) means that \8 collects everything else after the tab at the end of the seco

The replacement pattern for step #2 is \1\2\3\5\6\4\7\8\2\6\n

The replacement pattern will take the first example sentence, a tab, then the second e

Step #3: search for ti-ti and replace it with whitespace because it needs to be avail

For step #4, the search pattern is as follows: ( )(\t)(:+ [^\n\t]+)()(\t)([^\n\t]+)(\t

The replacement pattern is as follows: \2\3\1\5\6\4\7

( ) means that \1 searches for the followed by a whitespace, which can be found where

(\t) means that \2 searches for the tab before the first example sentence.

(:+ [^\n\t]+) means that \3 searches for and collects the one or more colons before th
\4 is the ta that separates the first and second example sentences.

\5 is the tab before the second example sentence.

([^\n\t]+) means that \6 will take the second example sentence.

(\t) is \7, which takes the tab after the second example sentence.

The replacement pattern is the following: \2\3\1\5\6\4\7

This takes the tab before the first example sentence, then the first example sentence,

Repeat steps 1-4 until the search pattern finds no matches. This will create sentence