Multiword Paraphrasing through Backtranslation for Automated Digital Content Management

Supervised by Luis Espinosa-Anke

Robert Michael Field
1941786

MSc Advanced Computer Science, Machine Learning
Cardiff University
27/11/2020

Abstract

Paraphrasing is an inherently human behaviour. To take an existing sentence, contextualize it and then re-word it so it retains meaning is a notoriously difficult task in machine learning. In a new digital age, where about two thirds of Americans get their news from online (Shearer and Matsa, 2018), the world relies more and more on web-hosted content. In this dissertation, the feasibility of using back-translation as a method for paraphrasing is investigated through experimentation using sentence similarity scoring metrics. The Jaccard similarity coefficient, ROUGE1 and ROUGEL metrics are used for these comparisons. In this project, three core romance languages (Spanish, Italian and French) are back-translated once and multiple times (known as "hops") and then scored against these three metrics. This dissertation proposes that the higher the score, the better the paraphrase - however, a score too high could be an exact copy of the original sentence and therefore not useful. This dissertation is made possible thanks to the generous co-operation of MarketMate (*MarketMate* n.d.) who provide the dataset of 10,000 rows for these experiments to be run on.

All experimental results and source code can be viewed at:
https://github.com/Monofoot/Multiword-Paraphrasing-for-Automated-Digital-Content-Management

Contents

1 Introduction

Natural language processing (NLP) is a subfield of computer science and artificial intelligence which found it's inception in the 1950s with Alan Turing's Turing Test (Turing, 1950) and subsequent thinking from John Searle's Chinese room paper (Searle, 1980). NLP has since evolved from slow, symbolic collections of rules to utilizing modern neural networks which are capable of artificial learning (Honnibal and Montani, 2017).

NLP is capable of manipulating sentence structures and words, using them to train models which can accurately predict aspects such as sentiment, tense and even synonyms (Araque et al., 2018), (Badjatiya et al., 2017).

One other element of NLP which is particularly interesting is that of back-translation (Poncelas et al., 2018), (Edunov et al., 2020). Back-translation is essentially the act of translating a corpus from language a to language b and then back to language a. It's a type of data augmentation which is essential when learning how to handle sensitive data which might lose a lot of meaning during the back-translation process. This loss of meaning can result in confusing sentences which bear little relevance to the original text.

However, this debauching of sentence structure can also be advantageous. With careful pruning and attention to the original and back-translated syntax tree, the back-translation process can produce sentences faithful to the original but with some metonyms and synonyms which offer a different flavour. For example, in figure 1 the message of the sentence is mostly the same but the adverb *badly* has changed to it's synonym meaning of *a lot*.

"The government's Test and Trace programme is struggling badly."
"El programa Test and Trace del gobierno está luchando mucho."
"The government's Test and Trace program is struggling a lot."

Figure 1: Poor back-translation

The result of figure 1 is a paraphrase of the original sentence which has resulted in some interesting data augmentation. Data augmentation makes the original sentence noisy, with some additional (but realistic) properties which are different from the original text (*badly* becoming *a lot*).

Back-translation is widely employed in industries which require cross-cultural communication, such as law or marketing, and is done (or at the very least approved) by humans and very rarely machines. This is because human interpreters have the capacity to understand shifts in emotional prose and cultural differences between the two languages. Training algorithms to accurately back-translate a corpus remains a challenging problem in the field of NLP, but advances are being made regularly (Edunov et al., 2020). An interesting analysis of some back-translated corpus might be the difference in similarity between the source and target language, or how many hops a sentence might need to increase or decrease it's accuracy - how many times can a sentence be back-translated and remain faithful to the original? Is this method of paraphrasing appropriate for digital content like blogs or news articles?

MarketMate (*MarketMate* n.d.), a company specialized in providing AI solutions for digital content creation, has kindly agreed to provide this dissertation with a corpus to answer these questions. The corpus they've sent contains 10,000 rows of scraped news articles hosted online, split into their headlines and summaries. The data is also provided in it's parsed format, with dependency nodes and parts of speech already labelled.

The topic of this project was discussed in a meeting held with MarketMate's CEO and the dissertation's supervisor. As this dissertation focuses on the practicality of paraphrasing in industry, it was crucial to make sure that MakretMate agreed with the direction this paper heads so that they might gain valuable insight into any patterns identified in their data. This project is both a greatly appreciated opportunity to work with an industry-proven dataset and also to benefit MarketMate with any interesting analysis.

MarketMate and the provided corpus are described at length in section 4.

1.1 Purpose and Research Question

The purpose of this project is to evaluate the syntactic parse trees of the original and back-translated sentences and conclude whether this is an appropriate method of paraphrasing. The amount of hops (back-translations) between language a and language b will also be taken into consideration, and languages will be scored based on how close to the original the back-translation can get. Research will be conducted to decide which scoring system is most appropriate when comparing syntactic parse trees and analysis of the sentences will also be carried out alongside this (such as word frequency, subject-verb-object relations and frequencies, how many of these have high scoring similarity).

Initially experiments will be ran across the three core romance languages of French, Spanish and Italian, with the capacity to test on other languages depending on how well initial experimentation performs (an interesting expansion to this analysis would be how well this scoring system works against more complex lexicons such as Japanese or Russian).

By exploiting syntactic irregularities (which lead to data augmentation) caused by back-translating, this project will investigate the feasibility of writing tree manipulation algorithms to rephrase sentences by augmenting them. This project also aims to answer the question of how much the sentence tree can be manipulated while keeping a high score of similarity - how much can a sentence be morphed before it loses all meaning or all resemblance to it's original form?

2 Systematic Literature Review

A literature review which isn't thorough and fair is of little value. In an effort to tackle any unconscious bias presented in this paper, a methodology for planning, selecting and reporting research papers needs to be selected. (Kitchenham and Charters, 2007)'s Guidelines for performing a Systematic Literature Review are a standard in software engineering and closely relates to other specialist fields. Because this paper balances on a line between software engineering and machine learning, guidelines with a degree of flexibility was key. As such, this paper's literature review is based mostly off of (Kitchenham and Charters, 2007)'s guidelines, which includes a predefined search strategy.

2.1 Planning

Before progressing with this systematic literature review (SLR), the project needs to be broken down into several questions which form the research protocol. These questions set the basis of an inclusion criteria for the SLR, meaning that if a paper is irrelevant or doesn't contribute to the question then it can be excluded. It's also important that these questions are meaningful and can lead to change in current practices and understanding. The aims and objectives of this project are discussed in section 1.1 and the following questions can be inferred from them:

Q1. What scoring methods are used when ranking generated sentences against a ground truth?

Q2. What techniques for graph comparison currently exist, and are they relevant to syntactic parse trees or dependency trees?

This research protocol has been carefully selected after identifying exactly what information needs to be researched to lead to a successful experiment.

Q1 concerns scoring methods, which need to be employed against original sentences from MarketMate's data and the back-translated paraphrases generated by the model. High scores should mean that the paraphrase is a close reconstruction of the original and low scores should likewise indicate a low similarity. The thought process moving forward is that by sorting these scores from highest to lowest, some insight could be garnered into what makes back-translation so successful or unsuccessful.

Q2 is essential for comparing the two syntactic trees produced by this project. The two trees (a ground-truth and the back-translated sentence) will have nodes and edges and the differences between them will highlight syntactic irregularities which are understandable by a machine. It's easy for humans to read two sentences and tell them apart, but the machine will need to compare the two graph structures. The research conducted on Q2 means qualifying papers must provide insight into graph comparison techniques used in regards to NLP.

2.1.1 Search Strategy

The search process must identify papers which qualify for the research questions listed above. This section of the project depicts how searches were conducted and whether or not papers qualify for inclusion.

The first stage of this search strategy is to first split the subject area into key words and synonyms to form search phrases.

back-translation	syntactic parse trees
graph comparison	paraphrasing
data augmentation	sentence scoring

Figure 2: Search Phrases

Figure 2 depicts the search phrases which are most relevant to the research questions.

The next stage hones in on what databases to use. There are several highly esteemed databases available, but because this project mixes between machine learning and computer science, extra databases need to be considered.

The three databases queried in for these search phrases were Cardiff University's online library portal, IEEE Xplore and ScienceDirect. The database from Cardiff University scans any database and journal it's subscribed to, so it actually returns quite a di-

verse array of results. IEEE Xplore and ScienceDirect are both industry-leading databases for computer science and technology and excellent resources for research. arXiv was later included in the search.

Searches were conducted with individual or combination variations of the search phrases from figure 2 - for example, "back-translation and data augmentation" or "back-translation and graph comparison". When a more refined search was required, or the results were far too many (in their thousands), more specific words were included. An ideal search would generate a maximum of 250 results per search, with anymore being excessive and impossible to scan through. A really important aspect of this search strategy is avoiding unconscious publication bias: this was achieved by using Cardiff University's online portal, which returned a mix of journals and databases which would require extra scrutiny. Extra care was also taken to scan the results of the papers before including them in the next stage of the SLR.

The search took place throughout September-October, with a second refined search taking place in November.

2.1.2 Study Selection

The study selection phase is when research is actually digested and ultimately added to the list of references based on inclusion and exclusion criteria.

For a paper to be considered for the SLR, it needs to provide evidence which answers the questions laid out in the planning stage. The paper is also passed through three stages of revision where several things happen:

In stage 1, papers are added liberally based on the title.

In stage 2, the introduction and abstract are read. If they're relevant to the research questions then they're included, but if the introduction or abstract don't answer the research questions then they're excluded.

In stage 3, the experiments and results are read, along with a scan of the contents of the rest of the paper. At this point, any papers which have been excluded are saved in an excel sheet so they might be accessed again in the future.

Stage	Papers	Additional Papers	Total Papers
Stage 1	67	5	5
Stage 2	19	0	5
Stage 3	10	2	17

Table 1: SLR Papers

Table 1 shows the stages in this study selection. A column for added papers was included because sometimes at that stage of the study a paper might immediately qualify for inclusion (though this often wouldn't be the case in stage 1 as papers are critiqued on title alone). During stage 1, 5 papers were found from external sources which were immediately added to the bibliography. These papers stem from a paper provided by the supervisor of this project.

During stage 3, the added papers during this step referenced other papers of interest. This led to the inclusion of two additional papers. New metrics for scoring systems were also identified, and as such additional searches were done for them which led to an increase in papers. These are discussed in the scoring section of the review.

All search stages were saved in an Excel sheet and can be replicated.

Syntactic Parse Trees

A syntactic parse tree is a tree representation of sentence structure. Sentences are comprised of many parts of speech, such as verbs, nouns and noun phrases. It's useful for both humans and machines to structure sentences in a format which is organized and easier to read. Syntactic parse trees are used frequently in NLP problems and are now the standard way to represent sentence structures (Galitsky, 2013).

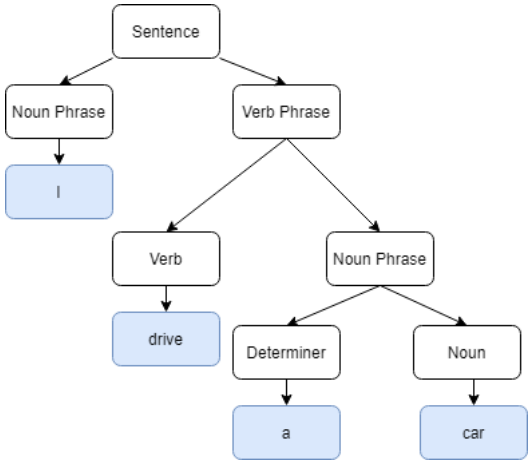


Figure 3: "I drive a car" syntactic parse tree

Figure 3 depicts the sentence "I drive a car". The words are split into their parts of speech and parsed across the tree.

Parse trees are graphs and as such can be assigned nodes and edges which allow them to be manipulated and analysed, producing interesting analytics - they can also be represented as parse thickets, which is how

(Galitsky, 2013) structure their attempt at parsing entire paragraphs as opposed to sentences.

These trees can also be compared against each other to find metrics such as accuracy or F-measure, which is what (Galitsky, 2013) propose with syntactic generalization. (Velardi et al., 2012) also compare two syntactic trees across their learned taxonomies by cutting the tree into hierarchical clusters, allowing them to analyse the tree from it's lower level (individual words with little meaning) to the higher levels (which provide them their ontology meaning).

Another paper which successfully weights two trees together and compares them is (Wang, Ming, and Chua, 2009). (Wang, Ming, and Chua, 2009) write a syntactic tree matching approach which find similar questions in Yahoo!'s search engine. Much like (Velardi et al., 2012), they also split the tree into branches and weight them.

Dependency Tree

Due to the nature of mathematical graphs, all leaves and branches can be represented numerically as nodes and edges. This means that other interesting methods of graph interrogation are available to this project, such as a dependency tree (Xu, C. Hu, and G. Shen, 2009) (Li, Zhu, and Zhang, 2011) which provides interesting insight into the relationship between parent and child nodes. A sentence always has a root word, and that root word has many children and some of those children have parents. Comparing dependency trees with a scoring metric discussed in the scoring research is equally as interesting as comparing the syntactic parse trees.

(Özateş, Özgür, and Radev, 2016) attempt this comparative approach to dependency trees by employing sentence similarity kernels for use in multi-document summarization analysis. (Özateş, Özgür, and Radev, 2016) also use Rouge-1 and Rouge-2 scores as a metric for defining their success which are further discussed in the scoring research.

There are several ways to represent dependency trees - spacy (Honnibal and Montani, 2017) use a horizontal plane and arrow arcs to represent their dependency hierarchy, whereas other applications use a standard tree (much like Figure 3).

Back-translation

Back-translation (Edunov et al., 2020), (Bojar and Tamchyna, 2011) is the process of translating from one source language to a target language and then back to the source language. This method of data augmentation was originally proposed for phrase-based machine translation in (Bojar and Tamchyna, 2011).

In an age dominated by online communication, users talking in other languages can often find it hard to understand each other (Aiken, 2002). This has led to a plethora of new translation tools, which are discussed at length with attention to secondary details by (Flournoy and Callison-Burch, 2003). This directly leads to more online tools and offline machine learning models for translation, making the process of back-translation quite easy and makes projects like this possible.

This method of translation produces a subform of language known as translationese, which (Toral et al., 2018) analysed extensively. (Toral et al., 2018) found that this translationese is actually easier to translate than original source languages, meaning potentially that the more hops a translation performs, the better accuracy it may have - it also means that back-translation for paraphrase generation is likely very accurate. (Graham, Haddow, and Koehn, 2020) claim that this hypothesis is incorrect because of their competing BLEU score, with (Edunov et al., 2020) stating that (Graham, Haddow, and Koehn, 2020)'s claim is actually inconclusive as it doesn't control for changes in content or source. The purpose of this dissertation is to evaluate whether this translationese is an appropriate form of paraphrasing.

(Poncelas et al., 2018) provide a solid analysis of current neural machine translation and evaluate the impact of using larger or smaller training sets.

Figure 5 is an example provided by (Miyabe and Yoshino, 2015). It's an example of poor translationese which would result in a low score. In this example some meaning has been lost in the translation (which is expected, as translating to any language using special symbols like Mandarin or Russian typically results in poorer scores).

Graph Comparison

Graph comparison is a sub-field of mathematics which is incredibly handy for machine learning and data analysis. Breaking graphs apart and analysing the various branches which form a tree can shine light on patterns and correlations between ordered datasets, which (Wills and Meyer, 2020) list in detail with shortest-distance measures and spectral distances and so forth. What's great about the nature of graph theory is that when it comes to comparing syntax trees in NLP, they're treated exactly the same as graphs in conventional mathematics: nodes may be labelled as a word in a sentence, but it's still a numbered node and still has a connected edge to another

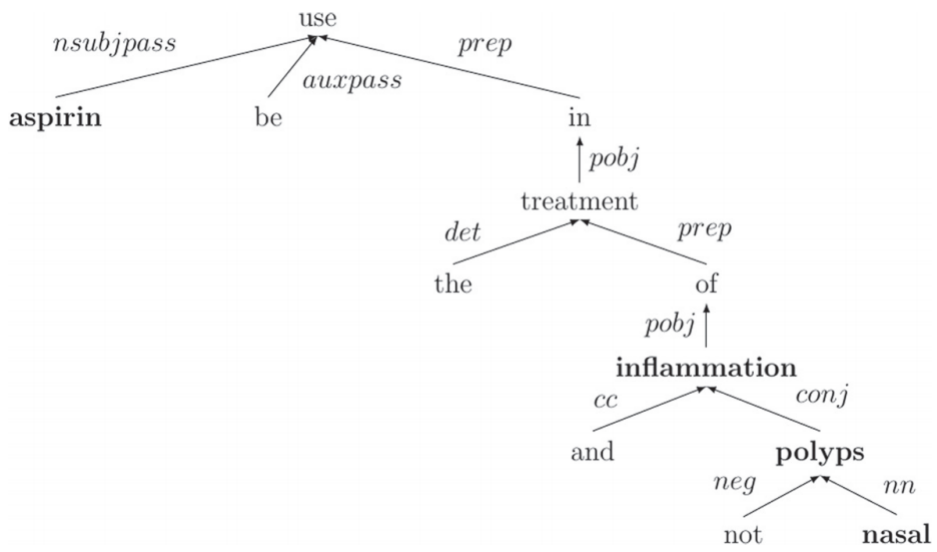


Figure 4: Dependency Tree ("Weissenborn, Schroeder, and Tsatsaronis, "2014")

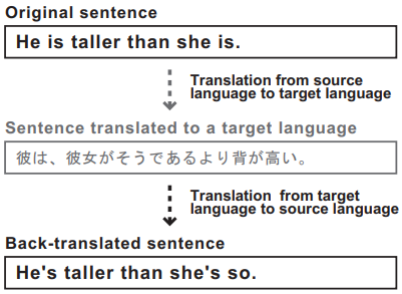


Figure 5: (Miyabe and Yoshino, 2015) example of back-translation

numbered node.

While searching for papers for graph comparison in this literature review, most of the topics relating to machine learning, graph comparison and NLP returned research on chemistry, medicine and DNA sequencing (such as (Melo and Daza, 2011) which offers interesting research into graph comparison but a different field). (Velardi et al., 2012) use various graph comparison techniques for their automatically learned taxonomy against a golden standard. They use a measure for cluster similarity which was introduced by (S. Wagner and D. Wagner, 2007). Clusters are sets which have been segmented into smaller sets, much like the notion of slicing branches off of a tree: as such, a cut can be represented as: $i(i \in \{0, ..., k-1\})$. Comparing graph structures based on their clusters gives the advantage of trimming massive data structures into smaller snippets, or allows analysis of a certain section of the tree disregarding all other relations.

The measure for comparing clusters of graphs previously mentioned by (S. Wagner and D. Wagner, 2007) was built upon the foundational work by (Fowlkes and Mallows, 1983). (Zager and Verghese, 2008) also makes some interesting observations on graph comparisons, but this paper is based on the neighbours in graphs and might not be wholly applicable to this dissertation. They propose to produce self-scoring similarity matrices between two graphs which are used on protein sampling.

However, (Zager and Verghese, 2008)'s work did reference (Blondel et al., 2004), who are very relevant to the dissertation and produce valuable insight into comparisons between two graphs for synonym extraction. (Blondel et al., 2004) do this to address the rising popularity of the web and the graph architecture it can take. They score their graphs based on the syntactic similarity between words. They build upon scoring metrics for matrix similarity introduced by (Kleinberg, 1999) and suggest that further research can be done in the field of similarity matrices using language.

Scoring

Scoring is the most critical element of both this literature review and the dissertation as a whole. There must be a quantifiable method for taking one sentence, back-translating that sentence, and scoring it based on the similarity of the original sentence. While searching for papers for this literature review, this research area was rich with literature and introduced

many new concepts to the project.

(Madnani, Tetreault, and Chodorow, 2012) present a wide overview of scoring metrics for paraphrase detection and is where the research really begins. Through the metric comparisons in (Madnani, Tetreault, and Chodorow, 2012), more papers relevant are added to the literature review.

BLEU was originally proposed in (Papineni et al., 2002). It's the most commonly used metric for evaluating translations between a source and target language, and works by calculating n-gram overlap between two sets of words. What this essentially means is that the more n-grams which exist in both the source and target language, the higher the score. (Madnani, Tetreault, and Chodorow, 2012) also mention that the BLEU metric has a penalty for translations which are too short. (Kravchenko, 2018) attempt to use BLEU as a scoring metric when translating from a Russain corpus, resulting in increased accuracy with a syntactically complicated language. (Kravchenko, 2018) cites (Madnani, Tetreault, and Chodorow, 2012), concluding that (Madnani, Tetreault, and Chodorow, 2012) used only a few machine translation examples, but their research uses only a BLEU score.

The BLEU score is calculated by counting the amount of matches, independent of their position in the sentence, in a candidate and reference text. The formula in figure 6 is discussed in length in (Papineni et al., 2002). Typically, BLEU is calculated based on N-grams, usually $N = 4$. The first figure 6 is the brevity penalty BP, which penalizes short translations. Let c be the length of the candidate corpus and r be the length of the reference corpus.

$$BP = \begin{cases} 1 & \text{if } c > r, \\ e(1 - r/c) & \text{if c} \leq \text{r} \end{cases}$$

Figure 6: BLEU Brevity Penalty BP

With the brevity penalty, the rest of the BLEU score can be calculated (Figure 7), where p_n is the average of the n-gram precisions and w_n is the positive weighting.

$$BLEU = BP \cdot exp(\sum_{n=1}^{N} w_n \log p_n)$$

Figure 7: BLEU Metric BP

BLEU is a strong contender for a scoring metric in this project. While a drawback of BLEU is that it only works with exact pattern matching and doesn't allow for synonyms (Madnani, Tetreault, and Chodorow, 2012), this shouldn't be an issue for this dissertation as the sentences being compared need to be compared as exact sentences.

(Edunov et al., 2020) also observe that the BLEU score is only really good when the source itself is a type of translationese. This, coupled with the criticism by (Madnani, Tetreault, and Chodorow, 2012) means that BLEU will not be used in this project as a scoring metric.

The METEOR metric (Denkowski and Lavie, 2014) is used for scoring translations against reference translations. It was originally designed to compete with and fix issues from the BLEU metric. In the paper from (Madnani, Tetreault, and Chodorow, 2012), they find that METEOR outperforms BLEU. (Banerjee and Lavie, 2005) also make a similar observation, stating that they received a score of 0.984 with human judgement in comparison to a BLEU score of 0.817.

The METEOR score is defined in (Denkowski and Lavie, 2014) and be defined mathematically in figures 8, 9 and 10. There are 3 steps to the METEOR equation. First, the R (weighted recall) and P (weighted precision) values must be calculated (figure 8). These values form the paramterized harmonic mean (figure 9) (Blair, 1979). A fragmentation penalty is then calculated to account for differences in word order (figure 10). Finally, the Meteor metric can be calculated (figure 11).

$$P = \frac{\sum_i w_i \cdot (\delta \cdot m_i(h_c) + (1-\delta) \cdot m_i(h_f))}{\delta \cdot |h_c| + (1-\delta) \cdot |h_f|}$$
$$R = \frac{\sum_i w_i \cdot (\delta \cdot m_i(r_c) + (1-\delta) \cdot m_i(r_f))}{\delta \cdot |r_c| + (1-\delta) \cdot |r_f|}$$

Figure 8: Calculating P (Precision) and R (Recall)

$$F_{mean} = \frac{P \cdot R}{\alpha \cdot P + (1-\alpha) \cdot R}$$

Figure 9: Calculating the parameterized harmonic mean of P and R Blair (1979)

$$Pen = \gamma \cdot \left(\frac{ch}{m}\right)^{\beta}$$

Figure 10: Calculating the fragmentation penalty

BLEU and METEOR are interesting metrics, but they really only provide a scoring avenue for translation quality. On the other hand, ROUGE (Lin, 2004) (Recall-Orientated Understudy for Gisting Evaluation) compares the actual summaries (sentences) instead. Much like BLEU, it works by comparing overlapping n-grams and returning a score from 1 to 0.

Another method of comparing sentence similarity is the Jaccard similarity coefficient. (Achananuparp,

$$Score = (1 - Pen) \cdot F_{mean}$$

Figure 11: Calculating the METEOR metric

X. Hu, and X. Shen, 2008) use this in an evaluation of sentence similarity methods, noting that the Jaccard coefficient is defined by the size of the intersection of the words compared to the size of the union of the words in both sets. This is a simple formula which operates on two sets of items to find how similar or diverse two competing sets are. The Jaccard coefficient can be notated as such:

$$J(A, B) = \frac{A \cap B}{A \cup B}$$

Figure 12: Jaccard Coefficient

While the Jaccard coefficient compares on the word level, (Abdullah, Sura, and Makttof, 2020) expand on this by instead comparing on the sentence level. This interesting research could be considered moving forward - the possibility of running the Jaccard coefficient on sentences as opposed to words is something which could be analysed in this dissertation's experimentation.

Because this dissertation is focusing on sentence similarity and not translation quality, the decision to move forward with a ROUGE score was made. Similarly, the Jaccard coefficient is also a good baseline metric to run against the machine translation. The ROUGE metrics are discussed further in the implementation and results section.

MarianMT

This section of the literature review was added after the SLR planning phase. This is an exceptional situation, as it addresses a problem which only became apparent during the design and implementation stage of the dissertation. During this stage, the chosen machine learning translation method was bandwidth limited and an alternative needed to be found. This is discussed further in section 3.2.2.

MarianMT Junczys-Dowmunt et al. (2018) is a machine learning framework written in C++ which can be used with the HuggingFace HuggingFace (n.d.) collection of pre-trained models. Run by Microsoft's translator research team and academic contributors, this is currently the engine behind all of Microsoft's translation services. Intel, Rakuten and the European Commission are some of many organizations which use this model for their translations.

MarianMT was ultimately selected as the model because it allows thousands of sentences to be run in parallel offline through CUDA (not accessing an API like the alternatives). It finds it's success in the neural networks Miceli Barone et al. (2017) and transformer models Vaswani et al. (2017) produced by academic contributors and has a large list of research papers which reference it.

3 Design

3.1 Requirements

In this section the project is broken up into manageable, defined steps. In the conclusion of this dissertation, this section will be referenced as the success criteria. Requirements for this project to be successful are on results gained by experiments discussed in section 1.1. This is dependent on the scoring metric used against back-translated paraphrases. This success criteria was chosen because the results gained by these experiments are quantifiable in the scoring metrics. If there are interesting results which need discussion then it means the experiments went well. It's also important to consider testing at the same time as implementation, meaning that well-written tests will also contribute to the success criteria and be critiqued in the conclusion (and discussed in implementation).

3.2 Technology Choices

3.2.1 Programming Language

When it comes to running experiments in machine learning (especially NLP) the immediate technology choice is the Python programming language. There are alternatives to this, such as the R programming language and JavaScript, but Python is much easier to use and has universal libraries.

The R Programming language is primarily used in statistics and graphics, meaning it's widely used by mathematicians and data miners who need to manipulate big data. JavaScript is always a good alternative to Python, but Python is the de facto machine learning language when dealing with libraries which are developed by other people.

Python was also the primary programming language because of the libraries required to run this project's experiments. For NLP, various functions such as part-of-speech tagging, tokenization of words and sentences and much of the preprocessing can be done through the the spacy library. Spacy is the industry standard in NLP and will be used to perform most of the tagging after sentences have been translated.

For reading and manipulating the csv data, pandas is the prime candidate. pandas allows csv files to be stored in dataframes which are easily accessed and changed, meaning preprocessing sentences and keeping the corpus organized is relatively easy.

While spacy provides the functionality to assign syntactic node and edge relationships, networkx is the library which builds the graph and stores the nodes and edges in a logical class. This class can be drawn to the screen or saved as images, which might lead to some interesting exploratory analysis. It also provides an easy way to interrogate the graph, by accessing the nodes and edges directly.

3.2.2 Translation Framework

There are several powerful translation frameworks available for Python. Before experiments can be conducted, there are certain criteria which must be met before a framework can be chosen. Firstly, the framework must handle a load of at least 5000 sentences in one experiment. It must also be actively maintained, with an active team of researchers contributing to it. There should also be a variety of languages available to translate to.

Searches immediately bring up the Google API (Google, 2020). This was initially promising, but upon running a test sample of several thousand sentences the Python interpreter was quickly blocked because of how many requests it made to the API. Google offers a solution through their Google Cloud subscription but a payment option wasn't feasible. After some more research, it became apparent that any API which uses an online tool quickly caps your requests, such as DeepL (DeepL, n.d.), which was the second choice.

Because of the limitations of an online tool it was decided to start looking for an offline pre-trained machine learning model. HuggingFace (HuggingFace, n.d.) has a collection of pre-trained models, and one of these is Microsoft's MarianMT (Junczys-Dowmunt et al., 2018). MarianMT is maintained by Microsoft's translation team and academic contributors. It uses neural networks (Miceli Barone et al., 2017) and transformer models (Vaswani et al., 2017) to do the translation and works completely offline, meaning there's no need to access an API. Preliminary tests went well, and the model was very easy to implement, making it the prime candidate.

The aim of this project is to use a single translation framework, though running the experiments through more than one could provide some interesting experimental results. (Hadla, Hailat, and Al-Kabi, 2015) do this and store the results of their experiments using both Google and Babylon with Arabic sentences. They concluded that Google's precision surpassed Babylon, proving that using more than one translation tool might be beneficial for this project.

3.3 Specification and Design

Because the implementation of this will be written in Python, this project has the opportunity to organize code in an object orientated framework. This is beneficial because it encapsulates functions specific for each part of the program and results in cleaner code which is easier to debug. As such, the rest of this design section assumes that the Python will be written with the object orientated programming paradigm. Python is powerful for machine learning because of it's implementations with abstracted libraries, and a lot of these machine learning projects are written as long scripts without object orientated programming in mind.

In the requirements section it was clearly stated that testing was paramount to the success of this dissertation. Testing for machine learning software is different from testing standard software (where a suite of unit and regression tests might suffice). The dataset needs to be split into a main and development set of 80% and 20% respectfully. This is because the dataset being used is 10,000 rows and spotting bugs early on will take a very long time. With a development set of 2,000 rows this doesn't become an issue. Further testing can also be done through the exploratory analysis stage, in which small sets of the corpus are analysed for anomalies which may need pruning.

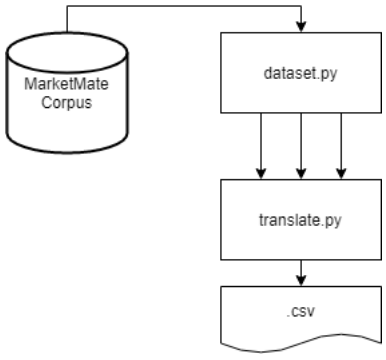


Figure 13: Flow Diagram

The flow diagram of the program is depicted in figure 13, and demonstrates how the corpus from the MarketMate data will be transferred into the first Python class. They're broken up into two classes so that the core functionality is abstracted from each

other: dataset handles the dataset, translate handles the back-translation. This abstraction allows for greater debugging and keeps errors confined to their class. It also means that because of the way this is designed, it's possible to have more than one class of dataset for a translation: it opens up the possibility of running numerous experiments in parallel in different languages. The results of the experiments are then stored in a .csv file for future analysis.

The dataset class should manage any preprocessing which needs to happen to the dataset. It also needs to perform functions which the translate class might want, such as only running experiments on n number of rows or getting specific information. As such, dataset also needs to have sufficient getters to return this information both to the translate class and to other functions inside of dataset. Figure 14 shows the basic UML diagram of what it should look like, but it's expected to include more functions when it comes to implementation due to the nature of mined data on the internet (extra pruning may or may not be required, for example).

Because the rows from MarketMate's corpus are long strings, they'll most likely need to be converted into literal objects to be parsed with NLP tools. This can be achieved with the literal evaluation functionality of ast (abstract syntax tree), though may need some extra work if it doesn't convert well enough (handling punctuation may raise problems).

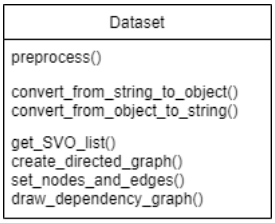


Figure 14: UML Diagram for Dataset

The translate class should handle all translation functionality. This class should only run with a dataset retrieved from the dataset class which has been sufficiently cleaned. In this translate class should be the option to specify how many hops to back-translate on this run and to calculate the appropriate scoring metric against those number of hops. At it's crux, the translate class needs to have a translation function which is malleable enough to take both English as a source language and any number of other languages as the target language (possible with spacy (Honnibal and Montani, 2017) and other machine learning frameworks).

While the translate class is running against rows, scores should be assigned to dictionaries alongside their original and translated languages. This acts as a reference and keeps all information about one entry tightly together, and it also allows averages and accuracy to be calculated across the length of the dictionary. This will need to be a nested dictionary.

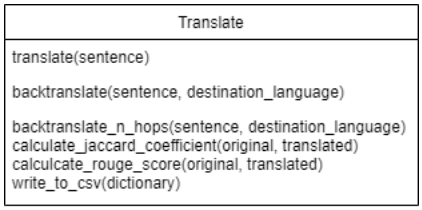


Figure 15: UML Diagram for Translate

The translate class also needs to calculate the ROUGE and Jaccard metrics. At each point in these functions, they should be appended to the relevant dictionary entry for the original and translated row.

4 Experimental Setup and Implementation

4.1 Dataset

The dataset for this project is kindly provided by MarketMate (*MarketMate* n.d.). It's a comma separated corpus of 10,000 rows of news titles and summaries scraped from the web.

This dataset is of particular interest due to the current pandemic. Because it has 10,000 news titles and summaries scraped from Q2 2020, most of the headlines are dominated by the phrases "COVID, Coronavirus, COVID-19". A word frequency analysis shows that synonyms of COVID-19 are in the corpus 1053 times ("COVID-19" turned up 316 times, "COVID" turned up 50 times, "coronavirus" 687 times - in comparison, "killed" was only in the corpus 37 times, "Trump" only 186 times). These terms shouldn't have an impact on paraphrasing as coronavirus isn't a new word and has translations in many other languages, but the compound form of COVID-19 may. This will be checked in the testing stage of this experimental setup.

(*MarketMate* n.d.)'s corpus is structured according to figure 16. The parsed_title and parsed_summary fields also include other valuable information as a list of lists. Figure 17 is a table mapping the tokens and their meaning.

title	Nightmare for four job scam victims
summary	PETALING JAYA: The promise of a well-paying...
parsed_title.	[["Nightmare",0,"PROPN","ROOT",0,"Nightmare "], ["for",1,"ADP","prep",0,"for "], ["four",2,"NUM","nummod",5,"four "], ["job",3,"NOUN","compound",4,"job "], ["scam",4,"NOUN","compound",5,"scam "], ["victims",5,"NOUN","pobj",1,"victims"]]
parsed_summary	[["PETALING",0,"VERB","amod",1,"PETALING"], ["JAYA",1,"PROPN","ROOT",1,"JAYA"], [":",2,"PUNCT","punct",1,": "], ["The",3,"DET","det",4,"The "], ["promise",4,"NOUN","nsubj",11,"promise "], ["of",5,"ADP","prep",4,"of "], ["a",6,"DET","det",10,"a "], ["well",7,"ADV","advmod",9,"well"]...

Figure 16: (*MarketMate* n.d.) corpus.

"Word"	Index	Part of speech	Modifier	Parent index	Parent
"Coronavirus"	1	"PROPN"	"nsubj"	2	"Coronavirus"

Figure 17: (*MarketMate* n.d.) tokenization.

4.2 Preprocessing

It's important that the titles used in the analysis is as true to the original as possible. With NLP, it's very easy to lose syntactic meaning of sentences by removing even the simplest trace of punctuation or a single word or letter. As such, no preprocessing on the word-level was performed and all titles were parsed in their original sentences.

However, after some exploratory analysis and initial results, it became obvious that there was some preprocessing required on the sentence-level. Firstly, there were a lot of duplicate titles in the dataset - this is most likely because of how the MarketMate data was crawled, picking up more than one website which uses the exact same titles. These duplicates were removed during preprocessing.

Some titles were also anomalous in that they were only one or two words long. A word frequency analysis performed on these very short titles shows that the most common violators was the word "FAQ", which is most likely accidentally being picked up when crawling for articles. Any titles which had less than or equal to 2 words were removed from the dataset.

This was the only preprocessing done. Lemmatization or stemming was a possibility, but these can drastically change the word meaning and structure of a sentence - they remained a possibility in case results were unsatisfactory, but ultimately weren't used. This is especially true with stripping punctuation.

4.3 Back-Translation

The back-translation is the primary driver for this experimentation. There aren't many restrictions in this section other than defining the input and output for sentences.

Because back-translation might be computationally expensive, all of the operations should take advantage of MarianMT's CUDA integration. CUDA allows code to be run in parallel, meaning that experiments could be reduced in time complexity from up to 12 hours to maybe 1 depending on chunk size. These experiments will be run on a NVIDIA 1080 ti, which will perform well with CUDA's parallel scheduling.

The back-translation function needs to be simple and scaleable for n amount of hops. Given the sentence as a string data type, it needs to translate that string into a target language and return the translated string. The implementation section will cover the algorithms used to accomplish this.

Results for each back-translation will be stored in a CSV file with the metrics discussed in the scoring research section. These scores will be averaged to give an overview of how the different language pairs perform. Because they'll all be stored in a nested dictionary, each individual translation can be observed. This allows for some extra analysis on individual rows - for example, selecting a random n amount of rows

or ordering by individual scores (ROUGE or Jaccard coefficient), the top n or bottom n of each of these scores and more.

4.3.1 Hops

Back-translation experiments can be re-translated using a process called hopping. When referring to hops in back-translation, it describes the amount of translations a sentence goes through.

For example, consider the sentence "Hundreds Sign-up to Virtual Royal Welsh Show in 24 Hours!". When translated to Spanish and then back to English, it returns the following translationese: "Hundreds register for the Virtual Royal Welsh Show in 24 hours!". This was a standard back-translation, which would be regarded as one hop. To make it two hops, the result (the translationese) is again translated to Spanish and then back to English, giving us a new paraphrase of "Hundreds Register - until the Virtual Royal Welsh Show in 24 hours!". This is now a two-hop back-translation, and can be iteratively expanded upon to become n-hops, with each having the chance to change the paraphrase in some way.

Part of the experimentation will be running the same set of back-translations on the MarketMate corpus, but evaluate them with n amount of hops. These will be stored in a separate CSV file which clearly indicates both the language and amount of hops: so a two-hop back-translation to Spanish will be labelled "en.es.en.es.en.csv".

4.4 Python Implementation

This section provides a complete tear-down of the Python implementation for this dissertation. The entire code can be found in the attached URL in the abstract. All Python implementation relates to the design in section 3. and aims to, where possible, stick closely to diagrams drawn there (though there are deviancies in areas which will be discussed). All Python code is written in compliance with the PEP 8 style guide.

4.4.1 dataset.py

dataset.py contains the Dataset class. This is where operations pertaining to the MarketMate corpus are performed.

The constructor for the dataset class does several things:

- Load the dataset using a pandas frame.
- Drop duplicate rows in the "title" column of the corpus.
- Drop rows which are less than two words.
- Convert the "parsed_title" column from a long string to Python objects.

Listing 1: dataset.py constructor

```
self.corpus = pd.read_csv('mscarticles.csv')

self.corpus.drop_duplicates(subset="title", keep="first",
      inplace=True)
for index, row in self.corpus.iterrows():
    self.corpus.loc[index, 'parsed_title'] =
    self.convert_from_string_to_objects(self.corpus.loc[index,
          'parsed_title'])
for index, row in self.corpus.iterrows():
    if self.is_less_than_two_words(self.corpus.loc[index,
          'parsed_title']) == True:
        self.corpus.drop(index, inplace=True)
```

An important aspect of the constructor here is the double loop over the corpus. This was necessary because each iteration edits the index location. If a row was less than two words and was removed from the corpus, then the converting from string to objects function would query a non-existent index location and fail. Trying to fix this by converting from string to objects before dropping the row didn't work because it increments or reduces the index based on it's location, causing that to error too.

Dropping duplicate rows was a built-in pandas function, but query if a row was less than two words wasn't. Listing 2 is the function which performs this check. It works by simply checking the length of the "parsed_title" row and returns true or false.

Listing 2: is_less_than_two_words

```
def is_less_than_two_words(self, data):
    """
    Remove entries which are less than two in length.
    """

    if len(data) <= 2: return True
    else: return False
```

This is the only preprocessing done. The constructor also takes a random title from the corpus which is used for some small analysis while implementing the code, but it has no use in any of the experiments.

There are many more minor functions in this class, like getters and setters, and these can all be viewed in the appendix or repository. The next interesting function to discuss is for extracting subject verb object (SVO) relationships. This was an important function to implement for initial analysis and future experimentation. It loops through each row and parses the words through spacy to be tagged. This

function also uses an external library from (NSchrading, 2015). This library works with a spacy implementation which synergizes well with the rest of this project. Extracting SVO was important because of some exploratory analysis performed during the experimentation stage - it's also useful for future work (like scoring sentence similarities for only rows which have SVO patterns). Listing 4 and 5 show the two functions used in the experimentation stage.

Listing 3: Extracting SVO patterns

```
def extract_subject_verb_object(self):
    """
    Extract subject verb object relationships.
    """

    nlp = en_core_web_sm.load()
    patterns = []
    for index, row in self.corpus.iterrows():
        indexed_title = self.corpus.loc[index, 'parsed_title']

        words = []
        title = []
        for token in indexed_title: words.append(token[0])
        title = self.convert_to_string(words)

        if type(title) is str: parse = nlp(title)
        elif type(title) is list: None # Just... do nothing...
        svo = findSVOs(parse)
        if svo:
            patterns.append(svo)
            self.list_of_svo_titles.append(indexed_title)
    return patterns
```

Listing 4: Total SVO

```
def total_subject_verb_object(self, list_of_svo):
    """
    Return the total number of subject verb object
        relationships.
    """

    return len(list_of_svo)
```

Listing 5: Most frequent SVO

```
def most_frequent_subject_verb_object(self, list_of_svo, max):
    """
    Find the most frequent subject verb object relationships.

    Note: using collections in this scenario requires the list
        to be
    mapped to a tuple.
    """

    counter = collections.Counter(map(tuple, list_of_svo))

    return counter.most_common(max)
```

In dataset.py the graphs, nodes and edges are all set. For any graph comparison between sentences to take place, the class needs to have access to the networkx directed graph utility. This is split into a few functions with different purposes - some are just for exploratory analysis, some for saving graphs as images and one important one, set_edges_and_nodes, which is what the translate class accesses when translating sentences.

Listing 6: Setting edges and nodes

```
def set_edges_and_nodes(self, data):
    """
    Create a directed graph and set edges and nodes.

    Takes a single sentence as input.
    """
    nlp = spacy.load("en_core_web_sm")
    parse = nlp(data)
    G = nx.DiGraph()
    labels = {}
    for index, word in enumerate(parse):
        G.add_node(word.i)
        G.add_edge(word.head.i, word.i)
        labels[index]=str(word.text) + " " + str(word.dep_)
    return G, labels
```

Listing 6 is the function which the translate class accesses. It creates a directed graph from a single sentence. Spacy is used to parse the word and set dependency hierarchies: edge relationships are based on the child parent connections. When a sentence is parsed for back-translation, the original passes through this function and so does the translated one. Both of these edges and nodes are stored in a dictionary for analysis. Figure 18 shows the drawn directed graph.

Listing 7: Drawing a directed graph

```
def create_digraph(self, data):
    """
    Create a directed graph.

    Save the output as an image.
    """

    G = nx.DiGraph()
    labels = {}
    plt.figure(figsize=(10, 10))
    for index, token in enumerate(data):
        G.add_node(token[4])
        G.add_edge(token[4], token[1])
        labels[index]=token[0] + " " + token[3]

    pos = graphviz_layout(G, prog='dot')
    nx.draw(G, pos=pos, with_labels=False,
        font_weight='bold')
    nx.draw_networkx_labels(G, pos, labels)
    plt.savefig("graphs/syntactic_parse_tree_en.png")
    plt.clf()
```

Listing 7 is an example of the various directed graph functions in the class. In this one, it's the same

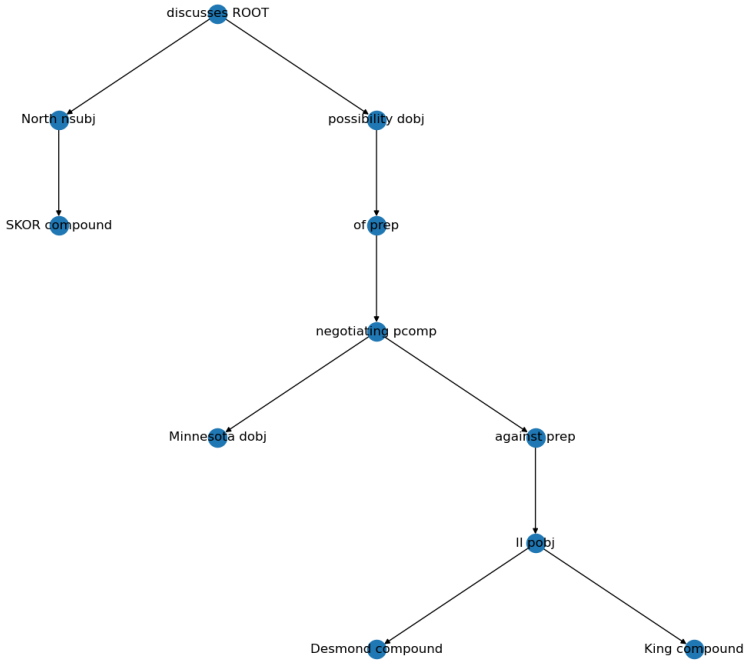


Figure 18: Generated dependency tree

as setting edges and nodes (simply making a directed graph), but this time it uses the matplotlib library to save the graph as a png.

Another function which offers insight into the sentence structures of the MarketMate corpus is to draw a dependency graph. This is a simple interface to spacy's displacy extensions - an example of one of the dependency graph is illustrated in figure 19.

Listing 8: Drawing a dependency graph

```
def draw_dependency_graph(self):
    """
    Draw a dependency graph, save as svg file in cwd.
    """

    nlp = en_core_web_sm.load()
    doc = nlp(self.get_literal_random_title())

    dependency_graph = displacy.render(doc, style="dep",
        jupyter=False)

    file_name = "graphs/dependency_graph.svg"
    output_path = Path.cwd() / file_name
    output_path.open("w",
        encoding="utf-8").write(dependency_graph)
```

4.4.2 translate.py

translate.py contains the Translate class. This is where all of the translation functionality is handled. It has references to dataset.py and instantiates an object of the dataset class which provides the sentences the be translated.

The most important function in Translate is the translation function. Listing 9 is the translate function - it uses a lambda function to set up a template which formats the data correctly for MarianMT to parse. MarianMt's documentation is thankfully quite detailed, so this implementation is relatively easy and didn't require much pruning from the examples they provide. The function encodes and then decodes the text in it's translated form.

Listing 9: The translate function

```
def translate(self, texts, model, tokenizer, language):
    template = lambda text: f"{text}" if language == "en"
        else f">>{language}<< {text}"
    src_texts = [template(text) for text in texts]

    encoded = tokenizer.prepare_seq2seq_batch(src_texts)

    translated = model.generate(**encoded)

    translated_texts = tokenizer.batch_decode(translated,
```

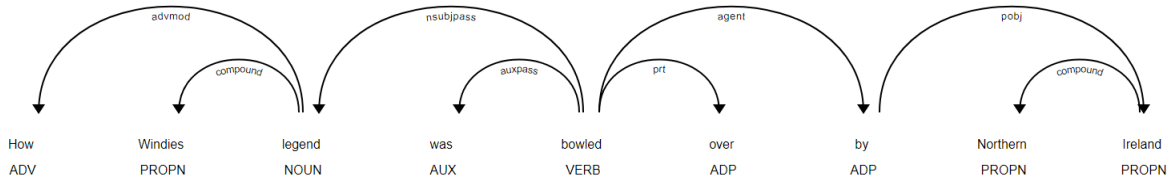


Figure 19: Generated dependency graph

```
        skip_special_tokens=True)
```

The back-translation function acts as a wrapper for the translate function. It takes an original and a source language as a string (these experiments will use "es", "fr" and "it"). There's a lot going on here, but what it's essentially doing is:

- For each row in the corpus:
- Create a directed graph for the original source language.
- Translate the original source language into the target language.
- Translate the target language back to the original source language.
- Create a directed graph for the new, back-translated language.
- Calculate the Jaccard coefficient.
- Calculate the ROUGE metric.
- Append all of these scores in a dictionary.

Listing 10: The back-translate function

```
def backtranslate(self, corpus, destination_language):
    """
    Translate to a language and then translate back.
    """

    for index, original_language in enumerate(corpus):
        title = []
        title.append(original_language)
        english_graph, english_labels =
            Corpus.set_edges_and_nodes(title[0])
        target_language = self.translate(title, target_model,
            target_tokenizer,
            language=destination_language)

        target_language_to_english =
            self.translate(target_language, en_model,
            en_tokenizer, language="en")
        target_language_to_english_graph,
            target_language_to_english_labels =
            Corpus.set_edges_and_nodes(tar...
        Jaccard_coefficient =
            self.calculate_Jaccard_coefficient(title[0].split(),
            target_language_to_english[0].split())
        self.list_of_Jaccard_coefficients.append(Jaccard_coeff...

        scores = self.get_rouge_metric(title[0],
            target_language_to_english[0])
        self.dict_of_translations[index] = {
            "Original": title[0],
            "Original edges": english_graph.edges,
            "Translated": target_language_to_english[0],
            "Translated edges":
                target_language_to_english_graph.edges,
            "Jaccard coefficient": Jaccard_coefficient,
            "rouge1 precision": scores['rouge1'].precision,
            "rouge1 recall": scores['rouge1'].recall,
            "rouge1 fmeasure": scores['rouge1'].fmeasure,
            "rougeL precision": scores['rougeL'].precision,
            "rougeL recall": scores['rougeL'].recall,
            "rougeL fmeasure": scores['rougeL'].fmeasure
            }
    self.write_to_csv(self.dict_of_translations)
```

The first score which is calculated is the Jaccard similarity coefficient. This has been discussed in the literature review, but it's essentially the size of the intersection of the words compared to the size of the union of the words. How many words overlap, and how many don't?

For this function, the parsed sentence is converted to lowercase strings because the Jaccard coefficient will penalize the sentence if translations are the same word but lack a capital letter. Because the corpus is a collection of news article titles, it makes sense that a vast majority of them use capital letters, but upon observing early results it becomes apparent that these capital letters are lost when translated.

Listing 11: The Jaccard similarity coefficient

```
def calculate_Jaccard_coefficient(self, original, candidate):
    """
    Calculate the Jaccard similarity coefficient.
    """

    original_lower_case = [word.lower() for word in original]
    candidate_lower_case = [word.lower() for word in
        candidate]
    intersection =
        len(list(set(original_lower_case).intersection(ca...
    union = (len(original_lower_case) +
        len(candidate_lower_case)) - intersection
```

```
    return round((float(intersection) / union), 2)
```

Calculating the ROUGE score is done using the rouge-score library (*rouge-score* n.d.). This was a simple interface which requires the original sentence and the back-translated sentence as the two parameters to compare. As discussed in the literature review, these experiments will use the ROUGE1 (unigrams) and ROUGEL (longest common subsequence) metrics. The rouge scorer library takes a candidate and compares against a reference.

Listing 12: The ROUGE metric

```
def get_rouge_metric(self, original_english, translated_english):
    """
    calculate the rouge metric.
    """
    scorer = rouge_scorer.RougeScorer(['rouge1', 'rougeL'],
          use_stemmer=True)
    scores = scorer.score(original_english, translated_english)
    return scores
```

4.5 Testing

Much of the testing for this project was done during immediate development with slices of the corpus. This was done by splitting the dataset in two: one smaller set for development. The small development set was exactly 20% of the corpus, so 2,000 rows, and was used whenever any debugging or testing needed to be done - this prevented having to run the entire dataset through the translation and preprocessing algorithms which was costly. Most of the output was printed to the terminal using Python and partial dictionaries were constructed to keep track of data.

System testing started once development testing ended (the program had satisfied the criteria to run experiments without error). This used the whole 10,000 rows in the corpus and ran from beginning to end with no issues.

The python code was run on a personal computer with a Nvidia 1080 ti GPU. It was not tested in a native Linux environment - it instead used anadonda to mimic a Linux environment.

As this is a mostly theoretical project, some testing was done through conversations with the supervisor and related work in papers. For example, it became apparent in the middle of the project that the Jaccard coefficient was incorrect because it was running on the edges and nodes of a sentence and not the words of the sentence itself - this was spotted by the supervisor because the numbers looked incorrect. An effort was also made to check numbers in previous papers in the literature review to see if results gained here are competitive at all with theirs.

5 Experiments

In this section the Python functions discussed in section 4.4 are run to create a dictionary of translated sentences with several scoring metrics. This section also discussed some initial exploratory analysis performed on the dataset before preprocessing was done: this was both for testing and analysis purposes, to see if anything needed to be changed.

5.1 Exploratory Analysis

Some initial exploratory analysis was performed on the MarketMate corpus to provide an insight into some of the patterns in the data. It's also important to do some exploratory analysis to identify problem areas such as duplicate entries or frequent patterns which don't quite make sense or require further explanation.

Analysis for this project focused mostly around finding subject verb object (SVO) relationships and the frequency that they show up. SVO relationships are widely researched in NLP and are efficient ways of identifying syntactic patterns in a dataset.

This project uses the (NSchrading, 2015) module to extract SVO relationships in the corpus. It's an easy interface to spacy which does most of the work behind the scenes, requiring minimum setup.

The result of the SVO pattern analysis were interesting and required a deeper look at the dataset to fully understand. Across the 10,000 rows, only 0.41% had SVO relations which were picked up by the parser.

SVO	Frequency
"news", "predicted", "lineup"	5
"microsoft", "acquire", "skills"	4
"landslide", "kills", "people"	4
"victoria", "records", "cases"	3
"man", "found", "dead"	3

Table 2: Test

Table 2 shows the top 5 SVO patterns. All of these SVO relationships require analysis of the corpus, and this was accomplished by searching for the unique terms (for example, "victoria" or "microsoft") in the CSV file. After searching these words and cycling through the results, it quickly becomes apparent why these SVO are so dominant.

"news", "predicted", "lineup" is actually a football headline. An example of a full title is "Manchester

City XI vs Liverpool: Confirmed team news, starting lineup, latest injuries for Premier League". Another title which triggers this SVO is "Watford XI vs Chelsea: Confirmed team news, predicted lineup, latest injuries for Premier League". This looks like a mistake by the parser, because what it's picking up is the second part of the title which is exactly the same for every football headline in the corpus.

"microsoft", "acquire", "skills" is a series of articles about Microsoft's pledge to help 25 million people acquire news digital skills in a COVID-19 economy. This is the first analysis which hits on information about COVID-19.

"landslide", "kills", "people" is several articles about the landslide at a jade mine in Myanmar which killed 100 miners. When MarketMate crawled their titles, this must have been a recent event.

"victoria", "records", "cases" is another COVID-19 hit. This was when Melbourne was beginning to shut down suburbs, so Victoria recording cases is, as expected, a big hit in the SVO analysis.

This initial analysis raises some interesting questions and has the potential for future research. SVO could be included in a future market evaluation, such as searching for MarketMate's most common patterns which could lead to changing future business decisions.

5.2 Back-translation

This paper proposes an experiment on the back-translation of news headlines provided by MarketMate. The experiments aim to answer the question of whether back-translation is an appropriate method for paraphrasing of automated digital content management and what the effect of several hops may have.

Some of the inspiration for these experiments and their layout are inspired by (Velardi et al., 2012)'s sentence comparisons.

These experiments are split into two parts with two purposes:

- For Spanish, Italian and French:
- Back-translate once
- Back-translate n times
- Score the original and translated sentence based on Jaccard similarity coefficient and ROUGE metrics

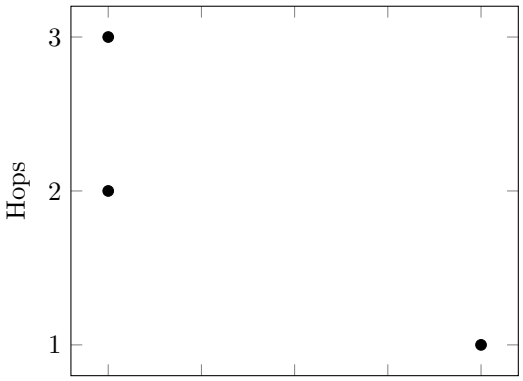
6 Results

All results are available in the repository as CSV files.

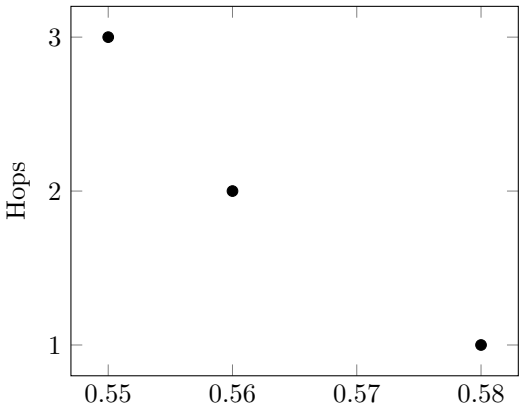
These sections break the results into their individual metrics and summarizes the findings. Also included is a small discussion on each metric with scores which are particularly interesting and may point to the cause of the translation.

6.1 Jaccard Coefficient

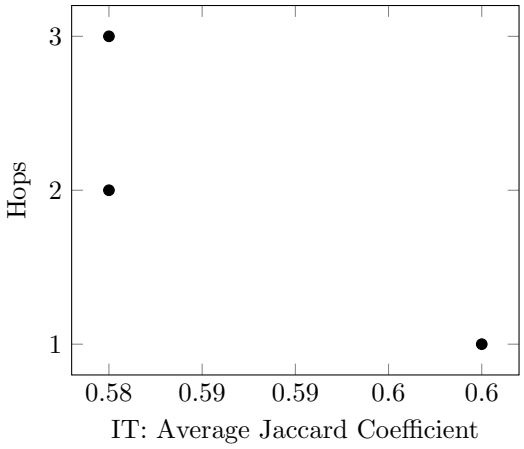
The results of the Jaccard similarity coefficient experiments sets the tone for what to expect with the other metrics. In all cases, a single back-translation hop provides the highest similarity score. When increasing the amount of hops to 2, almost no difference in similarity is discovered apart from the French translation. However, when the corpus is back-translated 3 times, there's a noticeable difference in similarity quality with a difference of 0.2 being recorded for each language. It's interesting that 3-hops will always result in exactly a 0.2 score reduction, demonstrating how confused the translationese can get.



ES: Average Jaccard Coefficients



FR: Average Jaccard Coefficient



6.1.1 Sample Score Discussion

Upon sorting the Jaccard coefficients from smallest to largest, there are quite a few interesting rows. For example, in the Spanish translation samples, figure 20 is a translation in hops 2 and 3. It wasn't possible to replicate this using the same back-translation with Google's online translator, meaning it must be an anomaly unique to the MarianMT model. This translation anomaly was exclusive to the Spanish corpus, which is odd considering that Spanish and Italian had the highest average Jaccard score.

Original: "Ne-Yo saves marriage"
Translated: "I-I don't–"
Jaccard coefficient: 0

Figure 20: Found in Spanish corpus with 2 and 3 hops

The French back-translation with 3 hops had the lowest score of 0.55. A lot of the lower-scoring sentences here were small sentences which had been inflated with hyperbole and words which poorly represented context (figure 21, 22).

Original: "Accenture to cut 900 UK staff"
Translated: "Focusing on the reduction of 900 employees in the United Kingdom"
Jaccard coefficient: 0.06

Figure 21: Found in French corpus with 3 hops

These low scoring sentences are very poor paraphrases of their original sentences and certainly wouldn't pass on any media platform. However, the higher scoring Jaccard coefficients were much more promising. A score of 1 means that exactly the same amount of words are in the translated sentence irrespective of word order.

Original: "Will ongoing shutdown eventually kill London's vibrant live-music venues?"
Translated: "Will the current stop end up killing the dynamic venues of live music in London?"
Jaccard coefficient: 0.04

Figure 22: Found in French corpus with 2 hops

High scoring is not always good, however. In the French translation corpus with 2 hops is figure 23 at a Jaccard score of 1. In this sentence, the words are exact but the order has been changed: it's implied in this paraphrase that FedEx is the name of the stadium, completely stripping the original meaning. This particular translation is translated the exact same way for every hop for every language. In fact, almost all of the sentences with a Jaccard score of 1 were exact matches of their original sentences (meaning that they're not really paraphrases, just exact copies).

Original: "Stadium sponsors FedEx ask Redskins to change name"
Translated: "FedEx stadium sponsors ask Redskins to change name"
Jaccard coefficient: 1

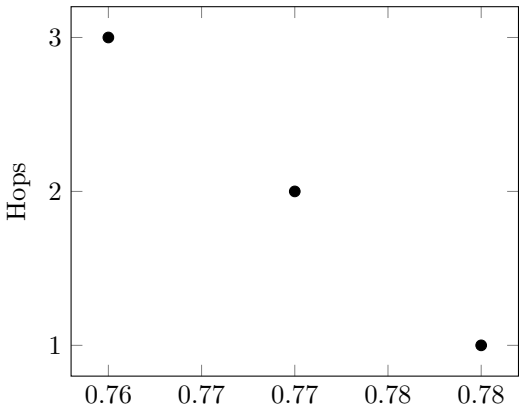
Figure 23: Found in French corpus with 2 hops

So then, perhaps the best paraphrases exist in the upper-medium of the scores. For example, figure 24 is a perfect paraphrase with a Jaccard score of 0.87. This sentence retains all of it's meaning and only changes one word, swapping out "injured" for "wounded". This translation would be a fine paraphrase for digital content.

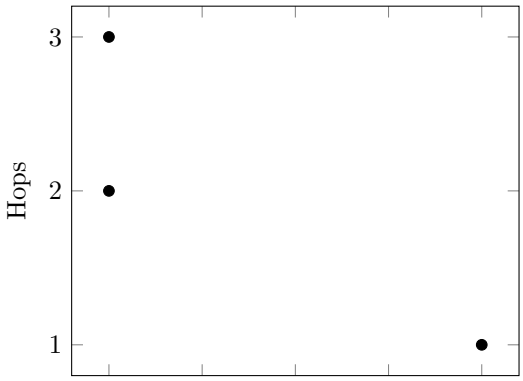
Original: "Man arrested on suspicion of attempted murder after injured police officer taken to hospital"
Translated: "Man arrested on suspicion of attempted murder after wounded police officer taken to hospital"
Jaccard coefficient: 0.87

Figure 24: Found in Italian corpus with 1 hop

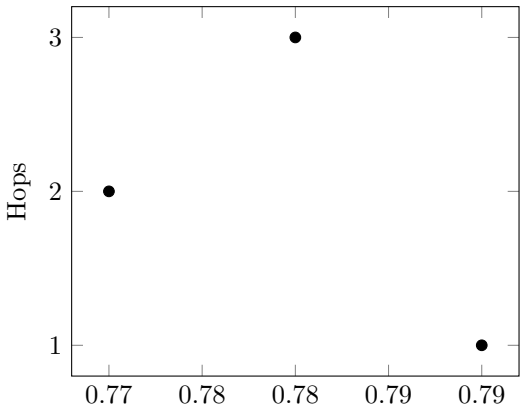
6.2 ROUGE1 Precision



ES: Average ROUGE1 Precision

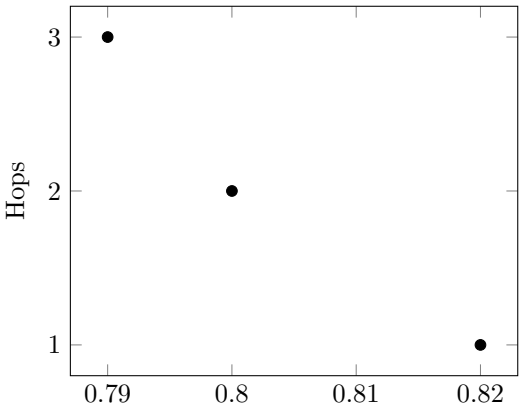


FR: Average ROUGE1 Precision

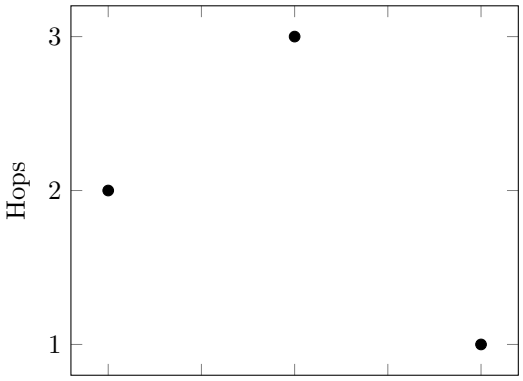


IT: Average ROUGE1 Precision

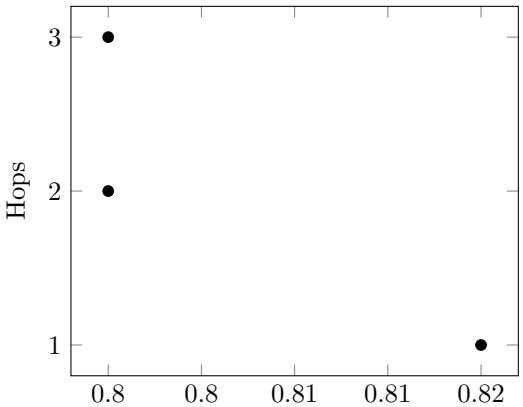
6.3 ROUGE1 Recall



ES: Average ROUGE1 Recall

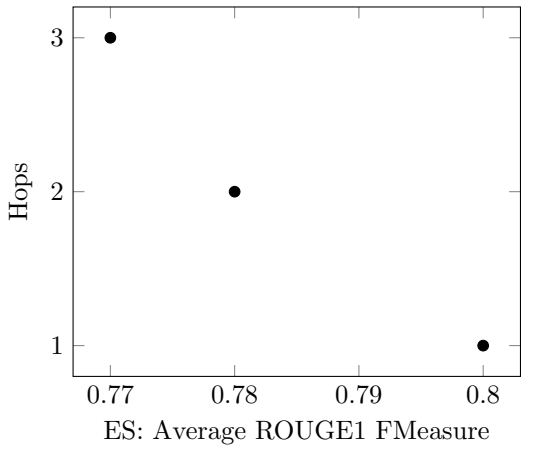


FR: Average ROUGE1 Recall

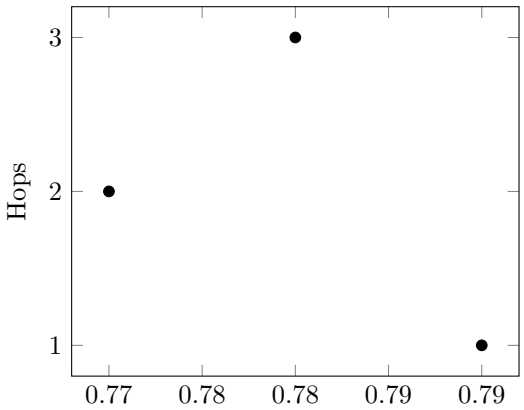


IT: Average ROUGE1 Recall

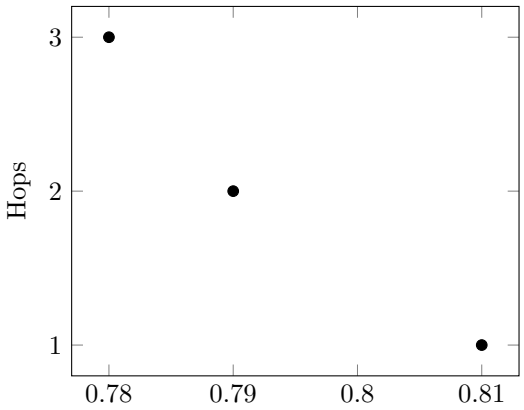
6.4 ROUGE1 FMeasure



ES: Average ROUGE1 FMeasure



FR: Average ROUGE1 FMeasure



IT: Average ROUGE1 FMeasure

6.4.1 ROUGE1 Sample Scores

ROUGE1 is the metric of overlap for each word between the candidate and reference text.

The ROUGE1 scores mostly mirrored the Jaccard metric. This is because they're both comparing unigram (one-word) sentence similarity. There a few cases in the corpi where the Jaccard score is 0 and the ROUGE1 is above 0 - for example, in figure 25. However, after scanning each corpus, these differences are extremely negligible and no anomalies provide interesting paraphrases.

Original: "Fantasy sports soar in lockdown period"
Translated: "Fantastic sport mounted at the time of locking"
Jaccard coefficient: 0
ROUGE1 Precision: 0.125

Figure 25: Found in French corpus with 3 hops

When the precision was high, it became apparent that this metric was much more forgiving than the Jaccard coefficient. This is evident in the scores across the hops, with precision ranging from 0.76 to 0.79 compared to the low Jaccard scores of 0.55 to 0.6. For example, in figure 26 the Jaccard coefficient was 0.33 while the ROUGE1 precision was 1. Figure 26 is another example of a score being 1 but being a poor paraphrase, as the entire meaning of the sentence has changed - in this particular sentence, it received a high score because the unigrams match.

Original: "California rolls back reopening of bars, restaurants and indoor venues"
Translated: "California reopens bars, restaurants and indoors"
Jaccard coefficient: 0.33
ROUGE1 Precision: 1

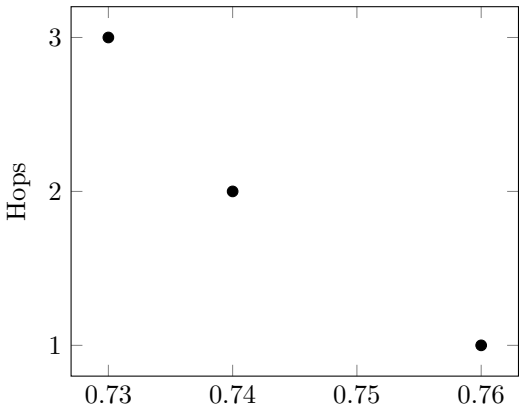
Figure 26: Found in Italian corpus with 1 hop

It stands to reason that if a score too low is unremarkable and a score of 1 is an exact copy, then just like the Jaccard coefficient, a medium-high score would produce the best paraphrases. This is proven in these experiments, as sentences with a score of 0.80 Jaccard similarity coefficient and ROUGE1 precision seem to produce the best paraphrases as demonstrated in figure 27.

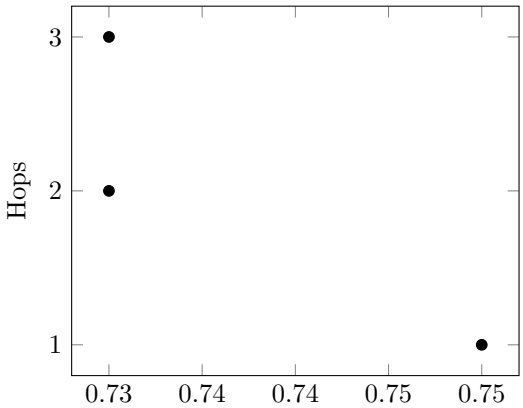
Original: "Video: Jordyn Woods hits the gym as Khloe & Tristan appear to rekindle"
Translated: "Video: Jordyn Woods hits the gym as Khloe & Tristan seem to rekindle"
Jaccard coefficient: 0.86
ROUGE1 Precision: 0.91

Figure 27: Found in Italian corpus with 2 hops

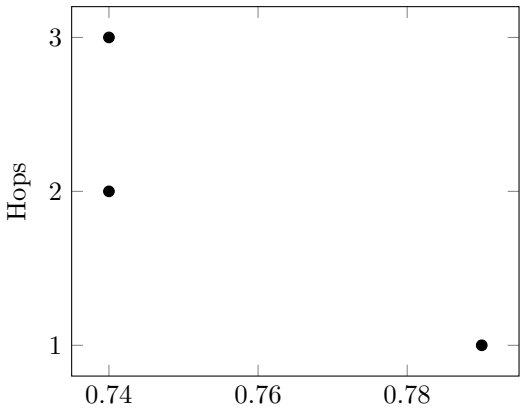
6.5 ROUGEL Precision



ES: Average ROUGEL Precision

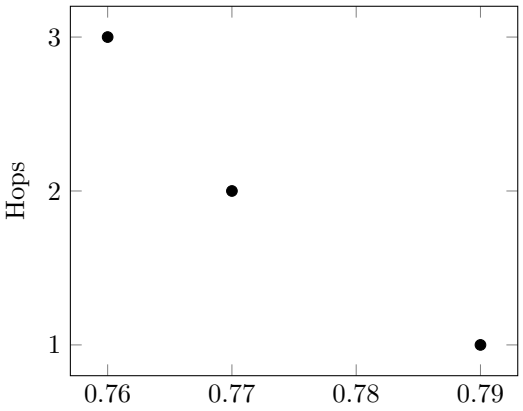


FR: Average ROUGEL Precision

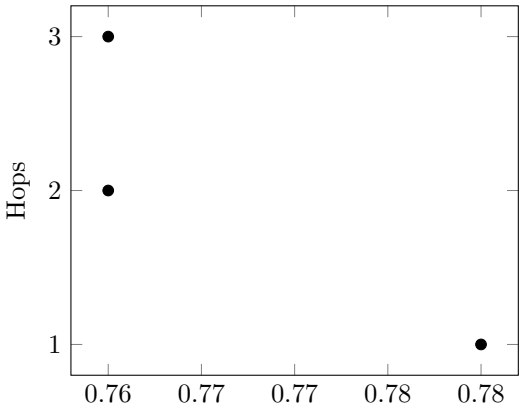


IT: Average ROUGEL Precision

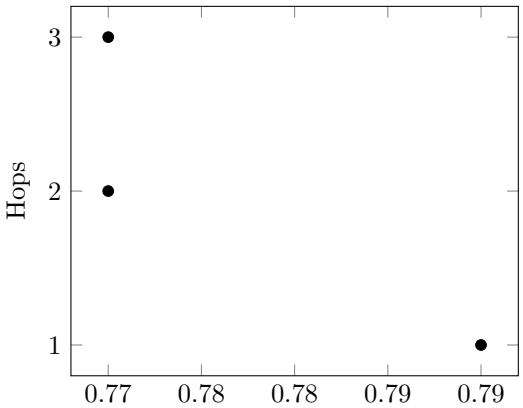
6.6 ROUGEL Recall



ES: Average ROUGEL Recall

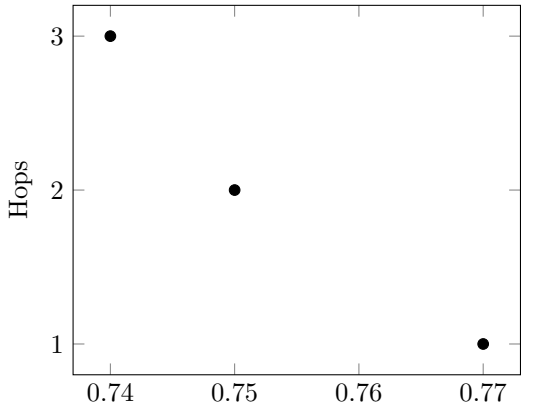


FR: Average ROUGEL Recall

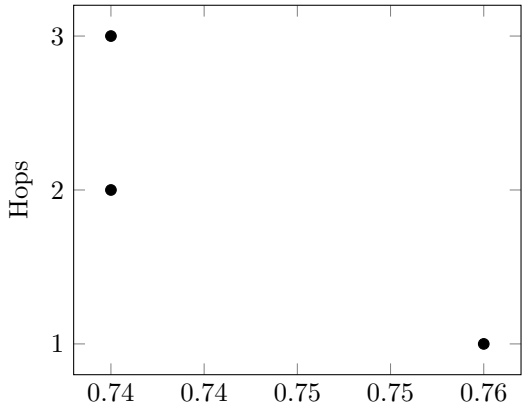


IT: Average ROUGEL Recall

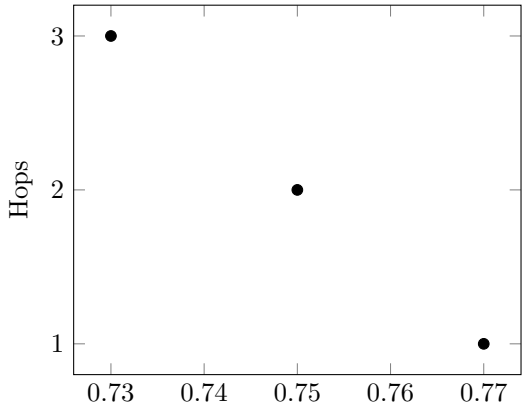
6.7 ROUGEL FMeasure



ES: Average ROUGEL FMeasure



FR: Average ROUGEL FMeasure



IT: Average ROUGEL FMeasure

6.7.1 Sample Scores

ROUGEL (Longest Common Sequence) takes the sentence structure and neighbouring n-grams to return a score from 0 to 1. In this experiment, running this metric alongside a Jaccard similarity coefficient and the ROUGE1 metric is interesting because it provides an avenue for some more intelligent paraphrasing.

Unexpectedly, the ROUGEL range of scores were actually lower than the ROUGE1. ROUGEL ranged from 0.73 to 0.79 in recall, whereas ROUGE1 ranged from 0.79 to 0.81. This is most likely because ROUGEL is more strict in it's scoring: ROUGEL scores by sentence structure and sequences of n-grams, whereas ROUGE1 simply looks for unigrams belonging in the candidate and reference sentences.

Similar to ROUGE1, ROUGEL shared a similarity of scores with the Jaccard coefficient metric but there were many sentences which had a score of 0 in the Jaccard metric but not ROUGEL. The same can also be said for very low Jaccard coefficients but competitive ROUGEL scores, as evidenced in figure 28. Figure 28 is also a very good paraphrase, coming in with a ROUGEL precision score of 0.7 but a Jaccard similarity of 0.33.

Original: "Bristol protests: Man arrested over pulling down of Edward Colston statue"
Translated: "Bristol Protest: Man arrested for removing Edward Colston's statue"
Jaccard coefficient: 0.33
ROUGE1 precision: 0.7
ROUGEL precision: 0.7

Figure 28: Found in Italian corpus with 2 hops

ROUGEL suffers from the same problem as ROUGE1 and Jaccard metrics in that too high a score is actually quite worthless for paraphrasing. For a paraphrase to be effective, it needs to be close to an original sentence but not exact, with a few synonyms swapped around and maybe some sentence structure changes (so long as the sentence makes sense). A score of 1 (see figure 29) means that the sentences are exact, or close to, with minor differences like grammar. Likewise, sentences with a score of 0 again give no valuable paraphrases as the resulting translationese is butchered beyond repair and unrecognisable.

Original: "How America's broken autopsy system can mask police violence"
Translated: "How America's broken autopsy system can mask police violence"
Jaccard coefficient: 1
ROUGE1 precision: 1
ROUGEL Precision: 1

Figure 29: Found in French corpus with 1 hop

It remains the notion of this experiment that the most useful paraphrases, then, are actually in the medium-high range of scores. Sorting the results by

highest ROUGE1, ROUGEL and Jaccard and selecting a sample from the 0.80-0.70 range returns a host of interesting, useful paraphrases. Figure 30 demonstrates this.

Original: "Video: China opens futuristic restaurant solely manned by robots"
Translated: "Video: China opens futuristic restaurant exclusively manned by robots"
Jaccard coefficient: 0.8
ROUGE1 precision: 0.89
ROUGEL precision: 0.89

Figure 30: Found in Spanish corpus with 3 hops

7 Conclusions

The results of this project's experiments conclude that with care and attention, back-translation might be an appropriate method for paraphrasing digital media content. This conclusion is based on the experiments with several similarity scoring metrics against a corpus of 10,000 sentences provided by MarketMate. Also included in the analysis were differing amounts of hops per language, with results showing that two hops can sometimes lead to an increase in paraphrase quality whereas any more than three hops will always result in a lower score.

The goals of this project laid out in the aims and objectives section are to evaluate the original and back-translated parse trees of sentences (namely their similarity) and conclude whether it's an appropriate method for paraphrasing, taking more than one back-translation hop into consideration. The goals were met by applying similarity metrics which were researched extensively in the literature review and selected based on what they score (for example, METEOR scored multiple translations which wasn't ideal). The Jaccard similarity coefficient, ROUGEL and ROUGE1 were chosen as the metrics the paraphrase attempts would be compared against.

There were initially some issues with selecting a translation framework to compute these paraphrases. During development the Google Translate API was used, but this was rate-limited as it accessed an online tool - the alternative was to pay for the service. This lead to a new section being added to the literature review which went over machine learning models for translation, as it was now a matter of making sure the model worked offline and wasn't capped by any bandwidth limitations. This resulted in MarianMT being selected as the model for translation.

The quality of the results lead to an interesting conclusion. Judging by results obtained in the experiments, it seems possible for back-translation to be used as a reliable means for paraphrasing sentences if the framework has a few restrictions in place, namely having a clamp function meaning sentences are picked from scores with a range of 0.70 to 0.90. This is because the most human-accurate paraphrases were in this range - any higher and they were exact copies, any lower and they were too dissimilar.

However, this might not be the case with the Jaccard coefficient metric. Results indicate that the Jaccard scores were always lower than the ROUGE scores in every case. There were several sentences which had Jaccard scores of 0 but high ROUGE scores of 0.7. This could possibly be wholly down to how it was implemented in the project and could potentially be enhanced by splitting out punctuation.

The experiment results across this project were consistent (with the Jaccard coefficient leaving room for improvement), and as such provides some realistic real-world paraphrasing analysis for digital media content.

7.1 Improvements and Further Work

Though the project went well and delivered some interesting insight into back-translated paraphrasing, there were a few hindrances encountered during implementation which could have been improved. For example, it's clear from the comparison of scoring metrics that the Jaccard coefficient is slightly too strict. Theoretically it should be close to the ROUGE1 precision, as both of these scores compare unigrams irrespective of their position in the syntactic tree. However, whereas ROUGE1 can achieve strong scores of up to 0.8, the Jaccard coefficients never go above 0.6. In any future work, a different implementation of the Jaccard coefficient will need to be tested: potentially lemmatizing or stemming words (potentially losing meaning), removing all punctuation (again, losing meaning).

MarianMT utilizes CUDA, a strong parallel computing platform which vastly reduces the time it takes to perform operations on lots of data. With 10,000 rows of sentences, it was important for this to work to reduce the time it takes to run experiments. While some efforts were made to write chunking functions (still present in the source code, just unused), none were implemented because of hardware problems at the time: the GPU running these experiments consistently froze after several hours of work. Any future would with this source code would require the chunking be fixed so parallelization can be fully taken advantage of.

A possible improvement of the experiments themselves could be filtering out sentences which are already in a different language to the source language of English. As these samples were mostly taken from the UK, some headlines were in Welsh - it seems that the MarianMT model did a good job at translating them all the same, but there are some very low scoring sentences which were originally in Welsh which weren't translated too well.

It might also an interesting idea to cross-reference some of these translations with a human translator, to verify their accuracy.

Using more than one translation model is a good idea, too. These could be used to compare the metrics across the different models (for example Google API, Yandex).

Because this project considers the feasibility of back-translation for paraphrasing, it thus makes sense that it could be further expanded into a framework for such a thing. A rough design might be a program which takes original sentences in English, back-translates them once, and then uses a min and max function (like a math clamp) to retrieve back-translations between this range. The range could use values discovered in this project (approximately around 0.80 for both ROUGE metrics).

References

"Weissenborn, D., Schroeder, M., and Tsatsaronis, G. ("2014"). ""Discovering Relations between Indirectly Connected Biomedical Concepts"". In: *"Data Integration in the Life Sciences"*. Ed. by H. "Galhardas and E. Rahm. "Cham": "Springer International Publishing", "112–119". ISBN: "978-3-319-08590-6".

Abdullah, S., Sura, M., and Makttof, M. (Mar. 2020). "Modifying Jaccard Coefficient for Texts Similarity". In: *Opcion* 32, pp. 2899–2921.

Achananuparp, P., Hu, X., and Shen, X. (2008). "The Evaluation of Sentence Similarity Measures". In: *Data Warehousing and Knowledge Discovery*. Ed. by I.-Y. Song, J. Eder, and T. M. Nguyen. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 305–316. ISBN: 978-3-540-85836-2.

Aiken, M. (Apr. 2002). "Multilingual Communication in Electronic Meetings". In: *SIGGROUP Bull.* 23.1, pp. 18–19. ISSN: 2372-7403. DOI: 10.1145/945541.945543. URL: https://doi.org/10.1145/945541.945543.

Araque, O., Gatti, L., Staiano, J., and Guerini, M. (2018). *DepecheMood++: a Bilingual Emotion Lexicon Built Through Simple Yet Powerful Techniques*. arXiv: 1810.03660 [cs.CL].

Badjatiya, P., Gupta, S., Gupta, M., and Varma, V. (2017). "Deep Learning for Hate Speech Detection in Tweets". In: *Proceedings of the 26th International Conference on World Wide Web Companion - WWW '17 Companion*. DOI: 10.1145/3041021.3054223. URL: http://dx.doi.org/10.1145/3041021.3054223.

Banerjee, S. and Lavie, A. (Jan. 2005). "METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgments". In:

Blair, D. C. (1979). "Information Retrieval, 2nd ed. C.J. Van Rijsbergen. London: Butterworths; 1979: 208 pp. Price: $32.50". In: *Journal of the American Society for Information Science* 30.6, pp. 374–375. DOI: https://doi.org/10.1002/asi.4630300621. eprint: https://asistdl.onlinelibrary.wiley.com/doi/pdf/10.1002/asi.4630300621. URL: https://asistdl.onlinelibrary.wiley.com/doi/abs/10.1002/asi.4630300621.

Blondel, V., Gajardo, A., Heymans, M., Senellart, P., and Van Dooren, P. (Dec. 2004). "A Measure of Similarity between Graph Vertices: Applications to Synonym Extraction and Web Searching". In: *SIAM Review* 46, pp. 647–666. DOI: 10.2307/20453570.

Bojar, O. and Tamchyna, A. (July 2011). "Improving Translation Model by Monolingual Data". In: *Proceedings of the Sixth Workshop on Statistical Machine Translation*. Edinburgh, Scotland: Association for Computational Linguistics, pp. 330–336. URL: https://www.aclweb.org/anthology/W11-2138.

DeepL (n.d.). URL: https://www.deepl.com/en/translator.

Denkowski, M. and Lavie, A. (2014). "Meteor Universal: Language Specific Translation Evaluation for Any Target Language". In: *Proceedings of the EACL 2014 Workshop on Statistical Machine Translation*.

Edunov, S., Ott, M., Ranzato, M., and Auli, M. (2020). *On The Evaluation of Machine Translation Systems Trained With Back-Translation*. arXiv: 1908.05204 [cs.CL].

Flournoy, R. and Callison-Burch, C. (Apr. 2003). "Secondary Benefits of Feedback and User Interaction in Machine Translation Tools". In:

Fowlkes, E. B. and Mallows, C. L. (1983). "A Method for Comparing Two Hierarchical Clusterings". In: *Journal of the American Statistical Association* 78.383, pp. 553–569. DOI: 10.1080/01621459.1983.10478008. eprint: https://www.tandfonline.com/doi/pdf/10.1080/01621459.1983.10478008. URL: https://www.tandfonline.com/doi/abs/10.1080/01621459.1983.10478008.

Galitsky, B. (Mar. 2013). "Machine learning of syntactic parse trees for search and classification of text". In: *Engineering Applications of Artificial Intelligence* 26, pp. 1072–1091. DOI: 10.1016/j.engappai.2012.09.017.

Google (2020). *Cloud Translation — Google Cloud*. URL: https://cloud.google.com/translate.

Graham, Y., Haddow, B., and Koehn, P. (Nov. 2020). "Statistical Power and Translationese in Machine Translation Evaluation". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Online: Association for Computational Linguistics, pp. 72–81. URL: https://www.aclweb.org/anthology/2020.emnlp-main.6.

Hadla, L., Hailat, T., and Al-Kabi, M. (Nov. 2015). "Comparative Study Between METEOR and BLEU Methods of MT: Arabic into English Translation as a Case Study". In: *International Journal of Advanced Computer Science and Applications (IJACSA)* 6, pp. 215–223. DOI: 10.14569/IJACSA.2015.061128.

Honnibal, M. and Montani, I. (2017). "spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing". To appear.

HuggingFace (n.d.). URL: https://huggingface.co/.

Junczys-Dowmunt, M., Grundkiewicz, R., Dwojak, T., Hoang, H., Heafield, K., Neckermann, T., Seide, F., Germann, U., Fikri Aji, A., Bogoychev, N., Martins, A. F. T., and Birch, A. (July 2018). "Marian: Fast Neural Machine Translation in C++". In: *Proceedings of ACL 2018, System Demonstrations*. Melbourne, Australia: Association for Computational Linguistics, pp. 116–121. URL: http://www.aclweb.org/anthology/P18-4020.

Kitchenham, B. and Charters, S. (2007). *Guidelines for performing Systematic Literature Reviews in Software Engineering*.

Kleinberg, J. M. (Sept. 1999). "Authoritative Sources in a Hyperlinked Environment". In: *J. ACM* 46.5, pp. 604–632. ISSN: 0004-5411. DOI: 10.1145/324133.324140. URL: https://doi.org/10.1145/324133.324140.

Kravchenko, D. (2018). "Paraphrase Detection Using Machine Translation and Textual Similarity Algorithms". In: *Artificial Intelligence and Natural Language*. Ed. by A. Filchenkov, L. Pivovarova, and J. Žižka. Cham: Springer International Publishing, pp. 277–292. ISBN: 978-3-319-71746-3.

Li, P., Zhu, Q., and Zhang, W. (2011). "A Dependency Tree Based Approach for Sentence-Level Sentiment Classification". In: *2011 12th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing*, pp. 166–171. DOI: 10.1109/SNPD.2011.20.

Lin, C.-Y. (2004). "ROUGE: A Package for Automatic Evaluation of Summaries". In: *ACL 2004*.

Madnani, N., Tetreault, J., and Chodorow, M. (June 2012). "Re-examining Machine Translation Metrics for Paraphrase Identification". In: pp. 182–190.

MarketMate (n.d.). https://www.marketmate.ai/.

Melo, J. A. and Daza, E. (2011). "On Molecular Graph Comparison". eng. In: *Current Computer Aided-Drug Design* 7.2, pp. 83–89. ISSN: 1573-4099.

Miceli Barone, A. V., Helcl, J., Sennrich, R., Haddow, B., and Birch, A. (Sept. 2017). "Deep architectures for Neural Machine Translation". In: *Proceedings of the Second Conference on Machine Translation*. Copenhagen, Denmark: Association for Computational Linguistics, pp. 99–107. DOI: 10.18653/v1/W17-4710. URL: https://www.aclweb.org/anthology/W17-4710.

Miyabe, M. and Yoshino, T. (2015). "Evaluation of the Validity of Back-Translation as a Method of Assessing the Accuracy of Machine Translation". In: *2015 International Conference on Culture and Computing (Culture Computing)*, pp. 145–150. DOI: 10.1109/Culture.and.Computing.2015.35.

NSchrading (July 2015). *NSchrading/intro-spacy-nlp*. URL: https://github.com/NSchrading/intro-spacy-nlp/blob/master/subject_object_extraction.py.

Özateş, Ş. B., Özgür, A., and Radev, D. (May 2016). "Sentence Similarity based on Dependency Tree Kernels for Multi-document Summarization". In: *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*. Portorož, Slovenia: European Language Resources Association (ELRA), pp. 2833–2838. URL: https://www.aclweb.org/anthology/L16-1452.

Papineni, K., Roukos, S., Ward, T., and Zhu, W. J. (Oct. 2002). "BLEU: a Method for Automatic Evaluation of Machine Translation". In: DOI: 10.3115/1073083.1073135.

Poncelas, A., Shterionov, D., Way, A., Buy Wenniger, G. M. de, and Passban, P. (2018). *Investigating Backtranslation in Neural Machine Translation*. arXiv: 1804.06189 [cs.CL].

rouge-score (n.d.). URL: https://pypi.org/project/rouge-score/.

Searle, J. R. (1980). "Minds, brains, and programs". In: *Behavioral and Brain Sciences* 3.3, pp. 417–424. DOI: 10.1017/S0140525X00005756.

Shearer, E. and Matsa, K. E. (2018). "News Use Across Social Media Platforms 2018". In: URL: https://www.journalism.org/2018/09/10/news-use-across-social-media-platforms-2018/.

Toral, A., Castilho, S., Hu, K., and Way, A. (2018). *Attaining the Unattainable? Reassessing Claims of Human Parity in Neural Machine Translation*. arXiv: 1808.10432 [cs.CL].

Turing, A. M. (Oct. 1950). "I.—COMPUTING MACHINERY AND INTELLIGENCE". In: *Mind* LIX.236, pp. 433–460. ISSN: 0026-4423. DOI: 10.1093/mind/LIX.236.433. eprint: https://academic.oup.com/mind/article-pdf/LIX/236/433/30123314/lix-236-433.pdf. URL: https://doi.org/10.1093/mind/LIX.236.433.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). *Attention Is All You Need*. arXiv: 1706.03762 [cs.CL].

Velardi, P., Navigli, R., Faralli, S., and Ruiz Martinez, J. M. (May 2012). "A New Method for Evaluating Automatically Learned Terminological Taxonomies". In: *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*. Istanbul, Turkey: European Language Resources Association (ELRA), pp. 1498–1504. URL: http://www.lrec-conf.org/proceedings/lrec2012/pdf/295_Paper.pdf.

Wagner, S. and Wagner, D. (Jan. 2007). "Comparing Clusterings - An Overview". In: *Technical Report 2006-04*.

Wang, K., Ming, Z., and Chua, T.-S. (2009). "A Syntactic Tree Matching Approach to Finding Similar Questions in Community-Based Qa Services". In: *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '09. Boston, MA, USA: Association for Computing Machinery, pp. 187–194. ISBN: 9781605584836. DOI: 10.1145/1571941.1571975. URL: https://doi.org/10.1145/1571941.1571975.

Wills, P. and Meyer, F. G. (Feb. 2020). "Metrics for graph comparison: A practitioner's guide". In: *PLOS ONE* 15.2, pp. 1–54. DOI: 10.1371/journal.pone.0228728. URL: https://doi.org/10.1371/journal.pone.0228728.

Xu, H., Hu, C., and Shen, G. (Jan. 2009). "Discovery of Dependency Tree Patterns for Relation Extraction". In: 2.

Zager, L. A. and Verghese, G. C. (2008). "Graph similarity scoring and matching". In: *Applied Mathematics Letters* 21.1, pp. 86–94. ISSN: 0893-9659. DOI: https://doi.org/10.1016/j.aml.2007.01.006. URL: http://www.sciencedirect.com/science/article/pii/S0893965907001012.