Imperial College London

BEng Individual Project

Department of Computing

Imperial College of Science, Technology and Medicine

Detecting Machine-Translated Documents

Author:
Prashant Gurung

Supervisor:
Dr. Thomas Lancaster

Second Marker:
Dr. Arthur Gervais

June 19, 2022

Abstract

The importance of translation cannot be overstated. Translation in literature preserves information, ideas, and provides accessibility to a wider group of readers. Translation in communication allows the exchange of information by overcoming the language barrier between two individuals.

With the introduction of machine-translation, translations can be performed swiftly on an input voice or text, and a sufficiently accurate result is produced. With state-of-the-art NMT systems being developed, the accuracy of these translations are constantly improving. However machine-translation can also be misused, with cross-language plagiarism being the most common case among students due to the accessibility of translation services.

Machine-translation in documents can be detected by professional translators, but does this hold when the evaluation is performed by a software? In this work, we create a system to detect machine-translation in documents. To achieve this, we explore several methods and implement them using natural language processing techniques to extract the relevant features from an input text. We evaluate the implemented methods using literary books provided by Project Gutenberg as the reference document. From our evaluation, we find the word-embedding method has the highest performance at feature-level on the same language pair with an accuracy of 73%. In addition, we find the most consistent and reliable method to be the dependency-tree method, with an average recall of 54% across different language-pairs and at document-level.

Acknowledgments

I would like to thank my supervisor Dr. Thomas Lancaster for his enthusiasm and invaluable insight throughout this project.

I would also like to thank my family and friends for their total support and belief in me.

Contents

Chapter 1

Introduction

1.1 Motivation

Translations of speech and writing are traditionally mediated by a professional translator that understands both languages. With constant developments to machine-translation systems such as Google Translate, the accuracy of their translation for both speech and writing is consistently improving. In addition, these systems support a wide range of language pairs, and are readily accessible.

Cross-language plagiarism occurs when a content by an author is translated to another language, and is claimed as their own work. It is a serious offence, and students caught plagiarising are expelled from their educational institution. Given the accessibility of machine-translation services, cross-language plagiarism has been pervasive among students. Additionally, multi-lingual students can perform cross-language plagiarism when reading and researching in one language and writing in another language. Finally, in the translation industry, freelancers hired to translate a document manually can cheat their contractor by using machine-translation.

Machine-translation can be detected by professional translators at the document-level [1]. However, substantial work and human effort is required in the case of reviewing hundreds of documents from students. In the case of contract cheating, the contractor may not be fluent in the translated language to notice the document has been machine-translated.

This brings motivation to the following problem: Are we able to identify with confidence when a work has been translated using a machine-translation service? If we can, we would achieve the following:

- Flag plagiarism in students coursework to be reviewed further for the marker without additional effort.
- Detect contract cheating if the freelancer has not authentically translated a document.

- Discourage students from using machine-translation services in order to cheat or plagiarise.

1.2 Objectives

We aim to create a system which, given an input text in English, performs analysis to classify whether machine-translation has been used. The following objectives for this project are proposed:

1. **Investigate which features of a text are relevant in detecting machine-translation**
   Investigate and implement past approaches of feature extraction in text to detect machine-translation.
2. **Evaluate which method performs best through a series of experiments**
   Using suitable evaluation metrics, determine which method performs best at feature-level and document-level in a fair evaluation.
3. **Extract relevant information from the input text**
   In the case that the document has been machine-translated, identify the key segments of the text that is machine-translated.

1.3 Ethical Issues

Overall, there are no significant ethical issues involved with this project. However, there are a few issues to consider.

The process of detecting machine-translation would require textual analysis, and therefore its contents to be read. However the document may contain personal or sensitive information. Therefore, the document should be managed securely, and removed as soon as it is no longer required.

Additionally, the detector could be misused. Students could spin the content in the document until it is no longer classified as a machine-translated text to work around being detected.

Deeper ethical issues arise when considering the impact on a student in the case when a human-written document is classified as `machine-translated`. However, it could be argued that professional translators and linguists are still the experts within the field. Therefore, they would still be able to identify cases of false positives [1].

1.4 Contributions

Our contributions in this project are the following:

- **Dependency Tree Method (Chapter 5)**
  We implement an approach which generates dependency trees for each sentence, extracting features from the dependency tree to detect machine-translation.
- **Word Embedding Method (Chapter 5)**
  We implement an approach which performs word embedding at paragraph-level, extracting features related to the word similarity between each embedding vector to detect machine-translation.
- **Back-Translation Method (Chapter 5)**
  We implement an approach which performs back-translation for each sentence, extracting features from the comparison of the source sentence to the back-translated sentence to detect machine-translation.
- **Implemented the word distribution method (Chapter 5)**
  We implement a method which determines whether a Zipfian word distribution is followed at document-level to detect machine-translation.
- **Evaluation on the following implemented methods (Chapter 6)**
  We perform a fair evaluation and compare the performance of each method to find which approach should be investigated further to reach better performance.

Chapter 2

Preliminaries

2.1 History of Machine Translation

Research into machine translation gained significant interest during the cold war. At the early stages, approaches in machine-translation produced poor quality translations, with the US ALPAC calling the attempts in its 1966 report *"expensive, inaccurate, and unpromising"* [2]. With the introduction of Statistical Machine Translation and recently Neural Machine Translation, the quality of translations and the number of supported language pairs have increased significantly. This section discusses the different paradigms in machine-translation that have been researched.

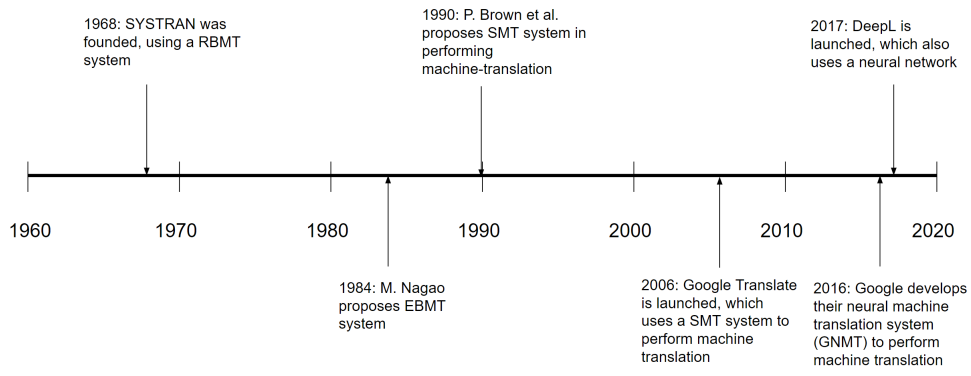


Figure 2.1: Timeline of significant events in Machine-Translation

Rule-Based Machine Translation

RBMT systems used a bilingual dictionary, and a set of rules describing the sentence structure of both language pairs to translate a sentence. The simplest approach following this paradigm was Direct Machine Translation, which would first translate word by word in a sentence, and then morph the sentence such that it follows the target languages grammar rules [3]. There were more complex approaches, but they all involved a dictionary, and a set of defined rules which the translation output

would have to follow [3].

The main problem with RBMT systems was that it required substantial human effort in first defining and maintaining a set of rules that would produce a good quality translation for a specific language pair [3].

Example-Based Machine Translation

An alternative to RBMT systems was the EBMT system proposed by Nagao. Translations in Japanese to English or vice-versa using RBMT systems were ineffective [4]. EBMT performed translations using previous translations which are similar to the given sentence as a basis and applying substitutions to the relevant sub-phrases.

English	**Japanese**
How much is that **big fish**?	Ano **ōkina sakana** wa ikuradesu ka
How much is that **blue car**?	Ano **aoi kuruma** wa ikuradesu ka

Table 2.1: Example of applying substitutions to the relevant sub-phrase

EBMT systems were very dependent on the example translations to perform accurate translations. For unfamiliar translations, which didn't have similar examples, the translations were not accurate.

Statistical Machine Translation

SMT treated translation as a machine learning problem [5]. A SMT system consisted of [6]:

- **Language Model**
  A language model, which scores a sentence S depending on its fluency and adequacy.
- **Translation Model**
  For a sentence pair (S, T), where S is in source language and T is in target language, a translation model assigns $P(T|S)$. This is the probability for how likely T is the translation for sentence S.

During training, the translation model uses the parallel corpus to assign $P(T|S)$. And during translation, a decoder searches for a viable translation by selecting that sentence in the source language for which the probability $P(S|T)$ is maximum [7].

The SMT paradigm gained popularity for commercial use, with Google Translate adopting a SMT-system to perform translation. The key reason was the training process was significantly shorter than previous systems, taking only a few days to train for a language pair [8].

Neural Machine Translation

The most recent approach to machine-translation has been NMT, which is based on using neural networks to learn a statistical model for machine-translation. There are different implementations of the neural network model, with encoder-decoder recurrent neural network being the state-of-the-art [9]. This architecture consists of an encoder neural network, which encodes an input sentence into a fixed-length vector, and a decoder neural network which generates a translation using the encoded vector.

This approach started to be preferred by most translating systems to SMT, with Google switching to using NMT systems for Google Translate in 2016 [10]. The NMT approach maintained the same advantages as the SMT system, whist producing an even better quality of translation.

2.2 Natural Language Processing

Natural language processing is the automatic manipulation of natural language, such as speech or text, by software [11]. This section explores the core techniques in natural language processing tasks relevant in this project.

2.2.1 Lemmatisation

Lemmatisation is the process where different inflected forms of a word is reduced to a single item called the lemma. For example, the words *"dance"*, *"dancing"* and *"dances"* are all reduced to its lemma *"dance"*. Lemmatisation decreases the vocabulary size of a task, allowing for faster computation [12]. More importantly, lemmatisation can clarify ambiguous words by taking the context of the word in the sentence before reducing to its lemma with the intended meaning.

2.2.2 Part-Of-Speech Tagging

POS tagging is the process where a word is assigned to its most probable POS tag using a neural network. The semantic meaning of the word, its surrounding context, and the pre-defined grammatical rules are considered during the assignment process [13]. For example, any word following immediately after *"an"* is likely to be a tagged as a noun.

The defined set of POS tags differs across the available NLP libraries. For example, the set of POS tags defined by the NLTK library [14] is different to the spaCy library [15]. The full list of POS tags for spaCy is shown in Appendix A, and for NLTK it is shown in Appendix B.

Token	POS Tag
Apple	PROPN
is	AUX
looking	VERB
at	ADP
buying	VERB
U.K.	PROPN
startup	NOUN
for	ADP
$	SYM
1	NUM
billion	NUM

Table 2.2: SpaCy POS Tagging: *"Apple is looking at buying U.K. startup for $1 billion"* [15]

2.2.3 Dependency Parsing

Dependency parsing is the process of analysing the structure of a sentence in order to identify dependency relations between words. Part-of-speech tags can be used with dependency relations to understand the structure of a sentence in greater detail. By performing POS tagging and identifying the dependency relationships existing in a sentence, a dependency tree can be generated. Each node in a dependency tree can only have a single parent node, but it can have multiple child nodes [16]. The defined set of dependency relations used by spaCy is shown in Appendix C.

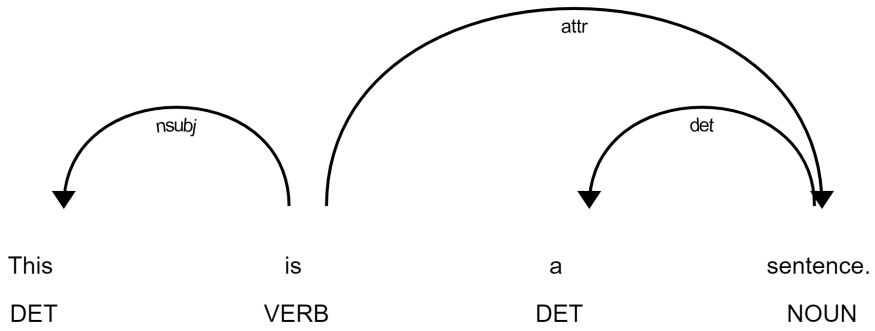


Figure 2.2: SpaCy Dependency Tree: *"This is a sentence."* [17]

Figure 2.2 shows us the generated dependency tree for the sentence *"This is a sentence."* using spaCy. The dependency tree highlights the dependency relations between words, and the POS tags assigned to each word. We can infer the following relations:

- The word *"This"* is the nominal subject of *"is"*.
- The word *"sentence"* is an attribute of *"is"*.
- The word *"a"* is the determiner for *"sentence"*.

2.2.4 Word Embedding

Word embedding is the process where a word is mapped to a vector of real numbers. The vectors represent semantic meaning of the word. When similar words are mapped to a vector space using a word embedding algorithm, the distance between the vectors is small. In fact, the smaller the distance, the more similar the words are. Conversely, the higher the distance, the more different the words are in their meaning [18].

Word2Vec

Word2vec [19] is a word embedding algorithm, which uses a large text corpus, and a feed-forward neural network to learn vector representations of a word. There are two approaches which both achieve this:

1. **Continuous Bag-of-words**
   For each word in the given text corpus, it's surrounding context words are one-hot encoded and averaged. The averaged vector becomes an input for the neural network, which produces a single output vector [20].

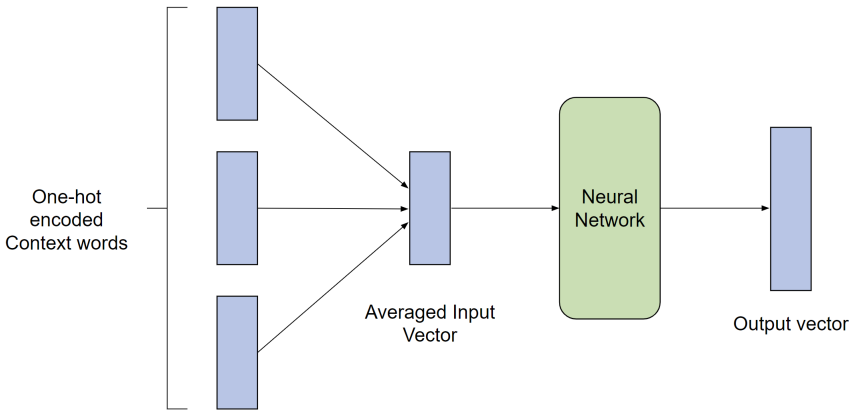


Figure 2.3: Continuous bag-of-words architecture

2. **Skip-Gram**
   Given an one-hot encoded target word, the neural network predicts multiple vector representation of context words that could be present for the target word [20]. Once trained, the vector representation for each one-hot encoded word is obtained by taking the hidden layer value of the neural network as the result.

GloVe

GloVe is a word embedding algorithm. A word co-occurrence matrix X of size $V \times V$ is constructed, where V is vocabulary size of the text corpus [21]. $X_{i,j}$ denotes the number of times word i has occurred in the context of word j.

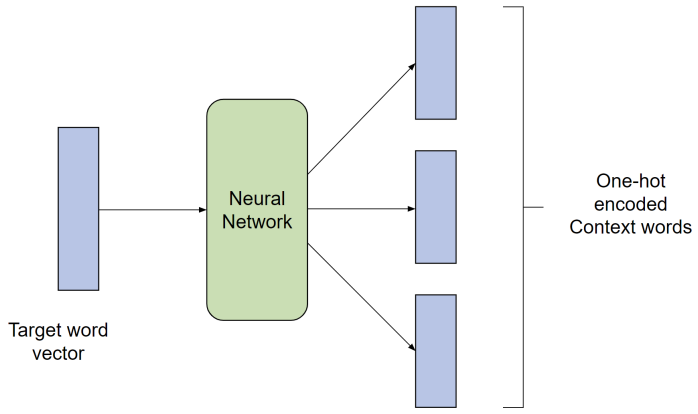


Figure 2.4: Skip gram model architecture

	the	car	drove	to	store
the	0	1	0	1	1
car	1	0	1	0	0
drove	0	1	0	1	0
to	1	0	1	0	0
store	1	0	0	0	0

Figure 2.5: Co-occurrence matrix for the sentence *"The car drove to the store"*

The main observation is the probability distribution derived from a word co-occurrence matrix can encode the semantic meaning of words. Table 2.3 shows the association of the words *"ice"* and *"steam"* to thermodynamics, and their disassociation to fashion.

Probability	**k = solid**	**k = gas**	**k = water**	**k = fashion**
$P(k\|ice)$	1.9×10^{-4}	6.6×10^{-5}	3.0×10^{-3}	1.7×10^{-5}
$P(k\|steam)$	2.2×10^{-5}	7.8×10^{-4}	2.2×10^{-3}	1.8×10^{-5}

Table 2.3: Co-occurrence probabilities for *ice* and *steam* trained on a 6 billion word corpus [21]

Once the co-occurrence matrix is constructed, it is used as input to a log-bilinear regression model. The model is then trained with weighted least-square objective to produce an embedding vector for a given word.

2.3 Neural Network Architectures

The field of Natural Language Processing has progressed further with development of neural network architectures. This section outlines the core ideas behind neural networks which are relevant in the task of detecting machine-translation.

2.3.1 Support Vector Machine

Support Vector Machine is a linear model which can be used for classification and regression tasks. The main goal of SVM is to identify the optimal hyperplane, which can separate a feature vector to its respective class [22]. This is achieved by finding the hyperplane which has the maximum margin between the support vectors. The support vectors are points which are closest to the hyperplane from both of the classes.

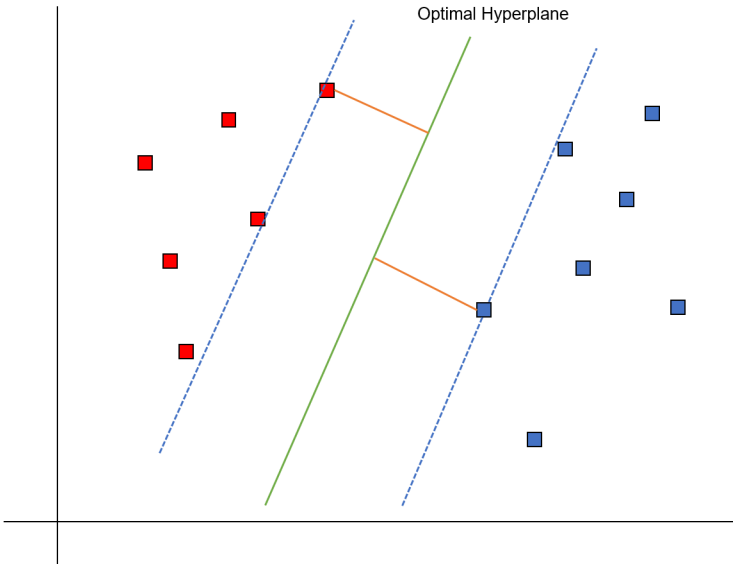


Figure 2.6: Example of an optimal hyperplane using SVM to separate two classes

In figure 2.6, the red squares indicate feature vectors which belong to a different class than the blue squares. The blue dotted line shows the support vectors for each class. The optimal hyperplane is identified by maximising the margin between the support vectors.

2.3.2 Convolutional Neural Network

CNN is a neural network composed of a sequence of convolutional and pooling layers. In the convolutional layer, multiple filters apply a strided convolution operation across the entire input vector. The convolution operation is a dot product between the filter and the selected window of input vector. In the pooling layer, a strided application of the pooling function is applied across the feature map to reduce its dimension size and improve computational efficiency [23].

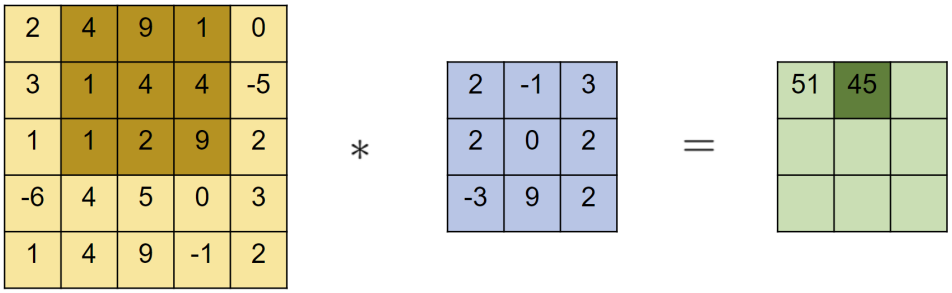


Figure 2.7: Example of a convolution operation

The sequence of convolutional and pooling layers forms a hierarchical structure, where the first layer extracts basic features, and subsequent layers extracts higher-level features. A fully connected layer is usually connected at the end of the sequence of convolutional and pooling layers.

CNNs have been commonly applied to computer vision tasks, such as image classification and object detection. However, they have began to show prevalence in NLP tasks such as sentiment analysis, sentence classification problems, and machine-translation [24].

Chapter 3

Related Works

This chapter explores past methods used to detect machine-translation in texts, where the translated document is only available. The key difference between the methods explored is which features are extracted from the given text.

3.1 N-gram Model

Machine Translation Detection from Monolingual Web-Text

Phrase salad phenomenon can be observed in translations performed by SMT systems using phrase-based models [5]. Phrase salad occurs in a sentence when each sub-phrase in the sentence is semantically and syntactically correct, but becomes incorrect when combined together. For example, phrase salad occurs in the sentence *"Although north wind howls, but sky still extremely limpid"* because *"north wind howls"* and *"sky still extremely limpid"* are both syntactically and semantically correct, but the merged sub-phrases results in an incorrect sentence.

In this project [25], Arase et al. explore features relevant to identifying phrase salads in SMT translated web-texts. They propose that phrase salads occur as a result of machine-translation. Therefore, identifying phrase salads will detect machine-translation in texts [25]. At least one of the three attributes are proposed to be present in a sentence containing phrase salad:

1. **Poor fluency between sequential phrases**
   In a phrase salad, the fluency is poor between sequential phrases.

2. **Poor grammar correctness between sequential and distant phrases**
   In a phrase salad, the tense and voice used between sequential and distant sub-phrases are inconsistent.

3. **Incomplete gappy phrases**
   In a phrase salad, there are incomplete gappy phrases. Gappy phrases are two or more distant sub-phrases occurring in the same sentence, which are incomplete without one another [26]. An example of gappy phrases are the

sub-phrases *"not only"* and *"but also"*, as a sentence only containing *"not only"* would be incomplete without *"but also"*.

To capture poor fluency between sequential phrases, two independent n-gram models L_H and L_{MT} are used to score the fluency of a sentence. L_H is trained with human-generated sentences and L_{MT} is trained with machine-translated sentences.

To capture grammatical incorrectness between sequential phrases, two independent n-gram models $L_{POS,H}$ and $L_{POS,MT}$ are used. $L_{POS,H}$ and $L_{POS,MT}$ are trained with part-of-speech sequences of human-generated sentences and machine-translated sentences respectively.

To capture grammatical incorrectness between distant phrases, two independent n-gram models $L_{F,H}$ and $L_{F,MT}$ are used. Using a POS tagger, sequences of function words are extracted from human-written and machine-translated sentences. Function words are words which sparsely appear in a sentence and are syntactically constrained in combination with one another. For example, the same preposition will not occur frequently in a human-generated sentence, but will in a sentence where phrase salad occurs. $L_{F,H}$ and $L_{F,MT}$ are trained with sequences of function words from human-generated sentences and machine-translated sentences respectively.

To capture incompleteness of gappy phrases, sequential pattern mining is performed in human-generated and machine-translated texts to collect gappy phrases. Two independent n-gram models $L_{G,H}$ and $L_{G,MT}$ are then trained on the sequence of the collected gappy phrases.

In total, eight separate n-gram models are used, each scoring a specific feature of the given sentence. Therefore, during feature extraction, a feature map of of size 9×1 is encoded with the extra element being the sentence length. To train a classifier in detecting machine-translation, a SVM classifier is used. A dataset of human-written and machine-translated sentences were used, and the SVM classifier was trained using the feature extracted sentences as input.

The method described in [25] can differentiate a bad human-translation to a SMT machine-translation. A bad human-translation can contain grammatical errors, but the structure of the sentence will make sense. This is not the case for phrase salads. However, the main drawback of this method is it assumes all machine-translated sentences contain phrase-salads. While the assumption is true for translations from SMT systems [5], it is unclear whether the assumption holds for NMT systems.

3.2 Back Translation

Back translation is the process of translating an input text to a target language, and translating the result back to its source language [27].

Detecting Machine-Translated Text using Back Translation

Vanmassenhove et al. observes that performing back-translation systems first time produces a low similarity between the translated and original texts. However, as more rounds of back-translation is performed, the similarity between texts increases due to infrequent words disappearing [28].

This project investigates the results in [28], and implements a system to detect machine translation in texts using back translation [29]. The following method was proposed to detect machine translated text:

Algorithm 1: Proposed schema for detecting machine-translated text

Data: source_sentence
Result: `human-written` or `machine-translated`

back_translation = back_translate(source_sentence);
f1, f2, f3, f4 = individual_bleu_score(source_sentence, back_translation);
f5, f6, f7 = cumulative_bleu_score(source_sentence, back_translation);
result = svm_classifier(f1, f2, f3, f4, f5, f6, f7);

return result;

The four features f_1, f_2, f_3, and f_4 are the individual n-gram BLEU scores between the source and back-translated sentence, where n ranges from 1 to 4. The remaining features f_5, f_6, and f_7 are the cumulative n-gram BLEU scores, where n ranges from 2 to 4. These features estimate the similarity between the input and back-translation, with a high BLEU score representing high similarity between the two sentences.

For the classifier, a SVM optimized with SGD is used. The dataset used for training were 2000 English-French sentence pairs from Europarl corpus [30]. The French sentences were machine-translated to English.

This method uses BLEU scores in calculating similarity between sentences. Calculating BLEU score is a quick and simple process. However, the BLEU score does not take the semantic meaning of words or the structure of the sentence into consideration when calculating the score. For example according to the BLEU score, the sentence *"I ate the apple"* is equally similar to *"I ate an apple"* as it is to *"I consumed the apple"* [31].

3.3 Word Distribution

Identifying Computer-Generated Text Using Statistical Analysis

Zipf proposed that the frequency of words in natural language can be modelled by an inverse relation to its corresponding rank in the frequency table [32]. Therefore,

the most frequent word in a document would occur approximately twice as often as the second most frequent word, thrice as often as the third most frequent word, and so on. Figure 3.1 shows Zipf's observation occurring in children's books.

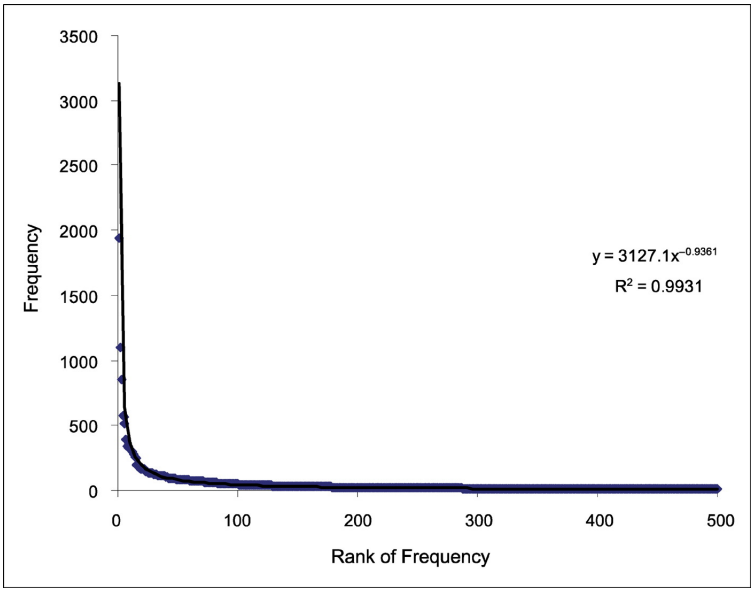


Figure 3.1: Frequency against rank of frequency for vocabulary in children's books [33]

In this project, Nguyen-Son et al. investigate whether Zipf's law can also be observed in machine-translated texts [34]. They extract frequency features, complex phrase features, and consistency features. The frequency features are extracted as follows:

1. **Linear Regression**
   Every word in the document is lemmatised to its corresponding lemma. The frequency of each word, and its rank against other words is collected. A regression line is then fitted for the frequency against rank at the log scale.

2. **Mean Squared Error**
   The mean squared error is calculated using the regression line as follows:

   $$MSE = \frac{1}{2N}\sum_{i=0}^{N-1}(y_i - f_i)^2$$

   In this equation, N is the vocabulary size of the document, y_i is the predicted frequency for a lemma at rank i, and f_i is the actual frequency for a lemma at rank i.

3. **Coefficient of Determination**
   The coefficient of determination is calculated using the regression line as follows:

   $$R^2 = 1 - \frac{\sum_{i=0}^{N-1}(y_i - f_i)^2}{\sum_{i=0}^{N-1}(y_i - \bar{y}_i)^2}$$

   In this equation, N, y_i, and f_i denote the same meaning as the MSE equation. The symbol $\bar{y}_i$ is the predicted frequency for a lemma at rank i using the mean distribution line $\bar{y}$.

From the following steps, gradient of the regression line, mean squared error, and coefficient of determination are extracted. The frequency features, along with complex phrase and consistency features are used as input to train a SVM classifier. The dataset used to train the SVM classifier were English and Finnish books collected from Project Gutenberg [35]. The Finnish books were machine-translated to English to be used as the machine-translated data.

When only considering the frequency features, this approach does not require to analyse the semantics of the document. To classify a document, only one feature extraction would need to be performed so the process would be fast. Additionally, Zipf's law can be observed in multiple languages, making it possible to extend the scope of this project to other languages.

However, this approach requires the input document to have a large vocabulary size and frequency of words to be effective. If the frequency of words is small, the fitted regression line becomes sensitive to any small change in the document.

3.4 Dependency Tree

This section explores different methods of feature extraction from a sentence by generating its dependency tree. The features extracted are used as input to a classifier to detect machine-translation in a sentence.

Predicting the fluency of text with shallow structural features: case studies of machine translation and human-written text

This project investigates which syntactic features correlate with the fluency of texts at sentence level [36]. They use Charniak-Johnson parser [37] to generate a dependency tree for a sentence. In their investigation, they found the following features strongly correlated with fluency:

1. **Sentence length**
   Shorter sentences were easier to interpret, so they were more fluent than longer sentences.

2. **Noun phrase length**
   Sentences with longer noun phrases were less fluent than sentences with shorter noun phrases. For example, the sentence *"[The dog] jumped over the fence and fetched the ball"* is more fluent than *"[The big dog in the corner] fetched the ball"* because its noun phrase is shorter.

3. **Average Verb phrase distance**
   The longer the average number of words separating two verb phrase, the less fluent the sentence became.

4. **Verb phrase length**
   Similarly to noun phrase length, the longer the verb phrase in a sentence, the less fluent it became.

5. **Tree depth**
   The longer the depth of the dependency tree, the less fluent the sentence became.

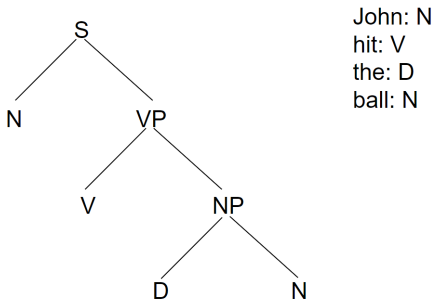


Figure 3.2: Example of a dependency tree for the sentence: *"John hit the ball"*

Additionally, their investigation used the features extracted to detect machine-translation in sentences. They found in their evaluation that the multi-layer perceptron was the most accurate classifier out of decision trees, logistic regression, support vector machines, and multi-layer perceptron.

A Machine Learning Method to Distinguish Machine Translation from Human Translation

Li et al. also explore a similar approach of using a dependency tree to identify machine-translated texts [38]. They observe that the parsing tree generated from human-written sentences are balanced, whilst parsing tree generated from machine-translated sentences are seriously right-deviated.

In their approach, they use a Stanford Lexicalized Parser [39] to generate the dependency tree. The following features for all constituent types and noun phrases from the dependency tree are then extracted:

- Number of right-branching nodes
- Number of left-branching nodes
- Number of pre-modifiers and adjectives before nouns
- Number of post-modifiers, and adjectives after nouns
- Difference between the right-branching nodes and left-branching nodes

- Difference between the number of tokens covered by right-branching nodes and left-branching nodes
- Difference between the pre-modifiers and post-modifiers
- Difference between the length in tokens of all pre-modifiers and all post-modifiers

The Europarl corpus [30] was used as the dataset. Four separate SVM were trained as the classifier with independent language pairs: French-English, German-English, Italian-English, and Danish-English.

One drawback of using features from a parsing tree is that it cannot represent texts at a higher level than a sentence. Therefore for a document, potentially thousands of parsing tree would need to be generated, which requires significant computation. However, the computation cost could be reduced by choosing a few sentences from the document, instead of analysing the entire document.

3.5 Word Embedding

Detecting Machine-Translated Paragraphs by Matching Similar Words

The coherence of a text refers to the sense relations between units such as sentences or propositions. Therefore, a coherent text is one that is logically and semantically consistent. Text analysis which measures coherence concerns itself with the construction and configuration of sense in the text, such as how its single constituents are connected so that the text becomes meaningful instead of being a random sequence of unrelated sentences and clauses [40].

This project develops a method to estimate coherence of a text at the paragraph-level to detect machine-translation [41]. They hypothesise that the coherence of human-written texts is higher than machine-translated texts, which is why coherence can be used to differentiate between the two. They propose the following:

1. **POS Tagging**
   POS tagging is performed for each word in a paragraph using the Stanford CoreNLP library [42].
2. **Word Embedding**
   Word embedding is then performed for each word using a GloVe model trained on the Wikipedia 2014 text corpus [21].
3. **Measure Euclidean Distance**
   Using the embedding vector, the euclidean distance between each vector is measured. If both vectors have the same POS tag, the shortest distance is kept. Otherwise, the distance for the POS tag pair is aggregated.

4. **Extract mean and variance**
   For each possible POS tag pair, the mean and variance of the aggregated distance is calculated. There are 1035 possible pairs POS tags defined by the Stanford CoreNLP library. Therefore, 1035 means and 1035 variances are calculated.

5. **Classify input paragraph**
   The means and variances calculated are used as the feature vector for an SVM classifier. The dataset used for training and testing was English and German transcripts from TED talks by native speakers. The German transcript was machine-translated to English using Google Translate, and every transcript was split into paragraphs.

This approach considers the semantics of the document unlike [34], [29], and [25] which only considers the syntax of the input text. In addition, text analysis is performed at a higher level. This approach performs text analysis at paragraph-level, whereas [25], [29], [36], [38] performs text analysis at sentence level.

3.6 Summary of Research

To summarise the related work discussed in this chapter, we have seen many different methods of feature extraction to detect machine-translation. The classifier used to detect machine-translation however, were all either support vector machine [25] [29] [34] [38] [41] or multi-layer perceptron [36].

It remains unclear however, out of all the methods researched, which method performs best in detecting machine-translation. This is due to the following factors:

1. **Dataset**
   Different datasets were used to train the classifier, and measure the accuracy of their method. For example, [38] uses the europarl corpus, while [41] uses TED talk transcript as their dataset.

2. **Translation**
   All methods use a machine-translation service to translate texts to be used as the `machine-translated` data during training and testing. However, the same translation service is not used. For example, [25] uses Google Translate with the SMT system, while [41] uses Google Translate with the NMT system. We do not know if a better quality of translation produced is harder to detect, in which case additional performance is gained from poor translations.

3. **Evaluation level**
   Our main objective is to be able to detect machine-translated documents, but none of the methods except [34] evaluate the accuracy of their methods at document level. It is either at sentence-level or at paragraph-level.

Therefore, we will implement the following methods: [29] [34] [36] [41]. The following methods were selected as they explore a new approach to one another. We will then perform a fair evaluation to find which features are relevant in detecting machine-translation at the document-level.

Chapter 4

Application Structure

In this section, we propose the design of our solution, as well as discuss the choice of language and technologies used.

4.1 System Design

The main interaction of this web app is to take an input txt file, perform classification according to the chosen method, and download a highlighted pdf which contains segments of text that are machine-translated. Figure 4.1 shows the main interaction between the client and server as a diagram:

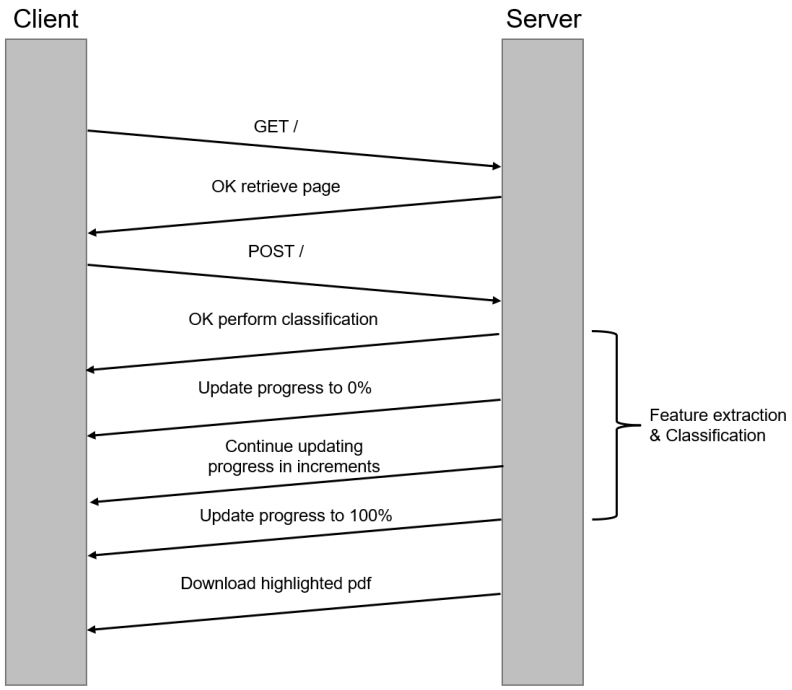


Figure 4.1: Interaction between client and server during feature extraction

When a user uploads a document and chooses a method, the back-end is responsible

for applying pre-processing techniques, feature extraction, and using a pre-trained model to perform classification.

4.2 Technologies

There are a number of different technologies and libraries which we have utilised throughout the project. In this section we provide an overview of their functionality as well as the justifications for using them.

4.2.1 NLP Libraries

SpaCy

SpaCy [15] is a python library, which uses state-of-the-art techniques to perform tokenisation, lemmatisation, dependency parsing, and POS tagging on an input text. It is simple to use, and it allows pipe-lining, and mini-batching to speed up NLP tasks.

NLTK

NLTK [14] is another leading library for natural language processing. Similar to spaCy, tokenisation, lemmatisation, dependency parsing, and POS tagging can be performed. However, the defined POS tags, and dependency tags are different to one another. In addition, NLTK has numberous pre-trained model to perform NLP tasks.

Hugging Face

Hugging Face [43] is a platform which provides access to state-of-the-art pre-trained ML and NLP models. Transformers to perform back-translations, models to perform POS tagging can all be accessed in this platform.

Sci-kit learn

Sci-kit learn library [44] provides a selection of efficient ML and statistical tools for a machine-learning tasks. For example in the classification task, the library provides SVM, Multi-Layer Perceptron, Linear Models, Decision Trees, and more that could be used as the classifier. Optimisation techniques to improve the performance of the classifier such as SGD, back-propogation, and more are also provided.

4.3 Web Deployment

We chose Python [45] over alternatives such as C++ [46] or Java [47] because it has an extensive NLP and ML library to choose from. Therefore, the past approaches in detecting machine-translation could be implemented using the available libraries

in Python. Choosing python left us with two choices for the web framework: Django [48] or Flask [49]. We chose the Flask framework, as it is lightweight and requires minimal code to get started. There is also no particular project structure enforced in Flask, whereas Django follows Model View Controller architecture.

Chapter 5

Project Implementation

5.1 Feature Extraction

This section details our implementation of the methods explored in Chapter 3. Our goal is to implement different methods of feature extraction of a given text, and evaluate which features are relevant in detecting machine-translation. Any design choices which deviates from how they have detailed their implementation is also discussed.

5.1.1 Word Embedding

We decided to implement the feature extraction described in [41] for an input paragraph. The key design choices to consider are:

- What do we use to perform word embedding on each word?
- What library do we use to perform POS tagging?
- What methods can be used to measure the word similarity between two embedding vectors?

In [41], GloVe is used to perform word embedding, the Stanford CoreNLP library is used for POS tagging, and the euclidean distance is used to measure word similarity between two embedding vectors.

We considered using word2Vec [19] and gloVe [21] to perform word embedding. We considered using the cosine distance and the euclidean distance to measure word similarity between two words. Finally, we consider using Stanford's parser and SpaCy's parser to perform POS tagging.

In our implementation, word similarity is measured with euclidean distance because it allows for a larger range in distance between vectors than cosine distance. Therefore, the difference of words is clearer. The word embedding is performed using GloVe trained on the Wikipedia 2014 corpus, and POS tagging is performed using

spaCy's parser.

Here is an pseudocode of our implementation in performing feature extraction of a paragraph:

Algorithm 2: Our implementation for feature extraction using word embedding

```
Data: paragraph
Result: means, variances

lemmatised_paragraph = lemmatise(paragraph);
pos_tagged_paragraph = pos_tag(lemmatised_paragraph);
pos_dict = init_pos_dict();
glove_dict = load_glove();

for (pos_pair1, pos_pair2) in permutations(pos_tagged_paragraph) do
    first_word = pos_pair1[0];
    second_word = pos_pair2[0];
    first_tag = pos_pair1[1];
    second_tag = pos_pair2[1];

    first_embedding = glove_dict.get(first_word);
    second_embedding = glove_dict.get(second_word);
    distance = euclidean_distance(first_embedding, second_embedding);

    if first_tag == second_tag then
        pos_dict[(first_tag, second_tag)] = min(distance, current_distance);
    else
        pos_dict[(first_tag, second_tag)].append(distance);
    end
end

means = [];
variances = [];
for (_, distances) in sorted(pos_dict.items()) do
    mean = get_mean(distances);
    variance = get_variance(distances);
    means.append(mean);
    variances.append(variance);
end
return means, variances;
```

5.1.2 Word Distribution

In [34], the frequency features were one of the 3 feature groups considered. They consider complex phrase features, and consistency features. However, we are only interested in finding out whether frequency features are relevant in detecting machine-translation. Therefore, we will only implement the frequency features described in [34].

There are no design choices to consider. However, it is interesting to find out whether using additional features that can be derived from the regression line which weren't described in [34] would aid performance.

In our implementation, we only consider the gradient of the regression line, the coefficient of determination, and mean squared error. Here is a pseudocode which performs feature extraction of a document. In this case, no additional metrics were added. However, the code would not deviate significantly in adding additional metrics.

Algorithm 3: Our implementation for feature extraction using word distribution

```
Data: document
Result: slope, r2, mse

texts_set = parse_book(document);
lemmatised_words_set = lemmatise(texts_set);

rank, freq_actual = collect_frequency(lemmatised_words_set);
rank = log10(rank);
freq_actual = log10(freq_actual);
linear_reg = LinearRegression();
linear_reg.fit(rank, freq_actual);
freq_pred = linear_reg.predict(rank);

slope = gradient(linear_reg);
r2 = r2_score(freq_actual, freq_pred);
mse = mean_squared_error(freq_actual, freq_pred);

return slope, r2, mse;
```

5.1.3 Back Translation

We decided to implement the feature extraction described in [29] for an input sentence. The key design choices to consider are:

- What do we use to perform back-translation?

- How do we measure the similarity between the source text and the back-translated text?

In [29], Google Translation is used to perform back-translation, and the BLEU score was used to measure the similarity.

In our implementation, we first used the googletrans [50] library to perform back-translation. This library is an API to Google Translate. However, in performing evaluation of several documents, we were blocked from sending too many requests. Therefore we used fairseq transformer model [51] to perform back-translation. To measure similarity between the two texts, we considered BLEU score.

Here is a pseudocode of our implementation in performing feature extraction of a sentence using back-translation:

Algorithm 4: Our implementation for feature extraction using back translation

```
Data: sentence
Result: f1, f2, f3, f4, f5, f6, f7

translator = load_fairseq_model();
first_translation = translator.translate(text, "en", "de");
back_translation = translator.translate(first_translation, "de", "en");

source_words = sentence.split();
bt_words = back_translation.split();

f1 = individual_bleu(source_words, bt_words, n=1);
f2 = individual_bleu(source_words, bt_words, n=2);
f3 = individual_bleu(source_words, bt_words, n=3);
f4 = individual_bleu(source_words, bt_words, n=4);

f5 = cumulative_bleu(source_words, bt_words, n=2);
f6 = cumulative_bleu(source_words, bt_words, n=3);
f7 = cumulative_bleu(source_words, bt_words, n=4);

return f1, f2, f3, f4, f5, f6, f7;
```

5.1.4 Dependency Tree

We decided to implement the feature extraction described in [36] for an input sentence. The key design choices to consider are:

- What parser do we use in extracting a dependency tree from a sentence?
- What constitutes a noun phrase and a verb phrase?

In [36], the Charniak parser [37] is used to extract a dependency tree from a sentence. However, the Charniak parser is no longer maintained and is outdated. Therefore, we have decided to use SpaCy library to extract the dependency tree from a sentence. We defined a noun phrase according to how spaCy identifies a noun phrase. However, SpaCy did not have a built-in identifier for verb phrases. In our implementation, a verb phrase is defined as any phrase which follows the pattern "`VERB ? ADV * AUX * VERB +`" when it is POS tagged.

Here is a pseudocode of our implementation in performing feature extraction of a sentence using dependency trees:

Algorithm 5: Our implementation for feature extraction using dependency tree

Data: sentence
Result: s_length, np_length, vp_length, n_np_length, n_vp_length, depth

```
dep_tree = generate_tree(sentence);
s_length = sentence_length(sentence);
np_length = unnormalised_np_length(dep_tree);
vp_length = unnormalised_vp_length(dep_tree);
normalised_np_length = np_length / s_length;
normalised_vp_length = vp_length / s_length;
depth = max_tree_depth(dep_tree);

return s_length, np_length, vp_length, n_np_length, n_vp_length, depth;
```

5.2 Classifier

Across the four different methods, a SVM classifier optimised with SGD was used. We used the SGDClassifier from the sklearn library [44] for this task. The number of epochs was set to 1000, and the stopping criterion was set to 0.001. The learning rate was set to the default value of the SGDClassifier.

There was consideration whether the number of epochs, stopping criterion, and other tunable parameters should be adjusted for each method until it reaches the best performance on the English-French dataset at feature-level. However, we decided against this approach because it would be using the test data as the training data. Additionally, it could potentially lead to overfitting for the English-French dataset, in which case the methods would not translate well across different language pairs.

Chapter 6

Evaluation

In this chapter, we evaluate the performance of the different methods implemented in detecting machine-translated documents. First, we explain the process by which we gathered human-written and machine-translated documents. Then, we cover the details on experiment design. Afterwards, we present a series of experiments that demonstrate the capabilities and limitations of our methods.

6.1 Dataset

The dataset gathered were literary texts and e-books provided by Project Gutenberg [35] written in English, French, German and Japanese. They were then divided into their chapters to represent a suitable length of a document. Alternatives such as the Europarl corpus [30] only consist of independently aligned sentences. However, a document is constructed of sentences and paragraphs which are dependent on one another, and use context. Therefore, a document is represented closer with a book. Additionally, the books provided by Project Gutenberg covers various themes and genres. The French, German and Japanese chapters were translated to English using Google Translate with the NMT system [52].

We selected French and German books because the languages both have similar linguistic distance to English [53]. Therefore, we can test our hypothesis that a trained classifier on the English-French language pair should be able to detect machine-translated documents from German to English with similar performance. Additionally, We selected Japanese books because its linguistic distance to English is much further than the French or German language [53]. Therefore, we can test whether it is easier to detect machine-translation in languages with further linguistic distance.

We chose Google Translate to translate non-English documents because it currently produces one of the most accurate translations using their state-of-the-art neural machine translation models [10]. Additionally, its popularity suggests that this is the first website that students attempting to plagiarise would use.

6.2 Experimental Procedure

This section details the followed procedures step-by-step in performing a fair evaluation for the different methods.

1. Feature Extraction

For all chapters, feature extraction is performed using the different methods. Each feature vector is then saved to its respective csv file. If it is a English or machine-translated French chapter, it is annotated with `human-written` for English chapters, and `machine-translated` for machine-translated French chapters. The German and Japanese chapters are not annotated as they are not used for training, and only used for testing.

2. Supervised Training

After feature extraction was performed, the representation of the two classes were balanced for each method by removing the excess feature vectors. We used a support vector machine, optimised with SGD as the classifier for the different methods, using the extracted feature vectors as input. For the evaluation to be fair, it is important that the same architecture for the classifier was used for different methods. Different classifiers will not have the exact same performance in this task, therefore additional performance may come from difference in classifier. After training, the classifier was saved as a pickle file to be used later for testing.

3. Testing at Feature level

Given an input document, different numbers of feature vectors are extracted from the methods during feature extraction. For example, the `Word Distribution` method collects frequency features of the document and produces one feature vector per document. However, the `Word Embedding` method performs feature extraction for every paragraph in the document, and so would produce p feature vectors where p is the number of paragraphs in the document. Similarly for the `Dependency Tree` and `Back Translation` method, feature extraction is performed on each sentence.

From the annotated feature vectors extracted, we use 80% of the feature vectors to train the classifiers for different methods. The remaining 20% of the feature vectors are used to test the accuracy, precision, recall, and f1 score. For each feature vector, the model will classify as either `machine-translated` or `human-written`, and since each feature vector is annotated, it is possible to find out whether it was correctly classified. For the machine-translated Japanese and German chapters, we used classifiers trained on all the feature vectors to predict the class. The recall for the machine-translated class was recorded.

Although this step does not represent the performance of the methods in detecting machine-translation at document-level, it is still useful to collect the result.

This is because a method which performs poorly in detecting machine-translation at document-level, but performs well at feature-level would indicate the problem lies in the algorithm used to classify the document, rather than the method itself.

4. Testing at Document level

We evaluate the performance of the different methods in detecting machine-translation at document-level for Japanese and German chapters. To classify a document as `machine-translated` or `human-written`, a majority vote of the classes is carried out. For example, a document with 100 feature vectors would need the classifier to predict 50 or more of the feature vectors as `human-written` for the document to be classified as `human-written`. The results from this step are the most realistic performance for each of these methods in detecting machine-translation, provided that the algorithm used to classify a document works well.

6.3 Quantitative Metrics

To reach appropriate conclusions from our experiments, it is vital that the methods are compared using suitable metrics. In this section, we discuss the strengths and limitations of several metrics that will be used.

Our experiments uses documents which are either entirely machine-translated or human-written. We do not consider documents which are partially machine-translated. Therefore, it is a classification task, and we do not need to consider metrics such as the mean-squared error which is used for regression tasks.

Accuracy

The accuracy is the proportion of features or documents which were correctly classified to its respective class. This metric works best when there is an even distribution of class across the features or documents.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}$$

Recall

The recall is the proportion of features or documents which were correctly classified `machine-translated` out of all the features or documents that were `machine-translated`. This is an important metric in finding out the performance of a trained classifier in detecting machine-translation on a different language-pair than its training set. For example, the performance of classifier trained on the English-French dataset on the German dataset, which allows us to better understand whether the trained models translate well across different language-pairs.

$$Recall = \frac{TP}{TP + FN}$$

Precision

The precision is the proportion of features or documents which were actually `machine-translated` from the features or documents that were predicted `machine-translated`.

$$Precision = \frac{TP}{TP + FP}$$

F1 Score

F1 Score is the harmonic mean of the precision and recall, and its an useful metric when there is an uneven balance of class distribution between `machine-translated` and `human-written` features or documents.

$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

6.4 Results

This section presents the results observed from performing the experiment according to the procedure defined.

6.4.1 Testing at Feature-level

English-French

In this experiment, feature extraction was performed on English and French chapters using the different methods. Table 6.1 shows that the word embedding method outperforms all the implemented methods across all metrics, with an accuracy of 73%. Word distribution has the second best accuracy at 66% but has a high precision and low recall. The back-translation method performed the worst out of all, with an accuracy of 56%. This was very likely due to the low number of features used for training and testing.

Detection Method	Accuracy	Weighted Precision	Weighted Recall	Weighted F1 Score	Test Features
Back Translation	56%	62%	56%	51%	921
Dependency Tree	59%	59%	59%	59%	96166
Word Distribution	66%	67%	66%	66%	401
Word Embedding	73%	73%	73%	73%	13784

Table 6.1: Evaluation metrics on the English-French dataset at feature level

German

In this experiment, the classifiers were trained on the English-French dataset, and its recall was measured on the German dataset at feature-level. This experiment was designed to find out whether the performance of a trained classifier has similar performance on a different language pair with similar linguistic distance to English.

Table 6.2 shows the performance of each method drops when using a different language pair as the testing dataset. The dependency tree method outperforms all the implemented methods with a recall of 55%. In addition, it is the method which drops the least when using a different dataset, with a recall drop of 4%. The word distribution method has the worst performance, and the drop in performance for the word embedding method is the highest.

Detection Method	**Recall**	**Correctly Labelled**	**Test Features**
Dependency Tree	55%	27181	49615
Word Distribution	36%	89	245
Word Embedding	46%	3134	6754

Table 6.2: Evaluation metrics on the German dataset at feature-level

Japanese

In this experiment, the classifiers were trained on the English-French dataset, and its recall was measured on the Japanese dataset at feature-level. This experiment was designed to find out whether machine-translation is easier or harder to detect from a language with further lingustic distance than the trained classifier.

Table 6.3 shows the performance of each method drops in detecting Japanese dataset at feature level than the German or English-French dataset. The word distribution method has almost no performance, with a recall of 10%. The dependency tree method drops performs the best and has the least drop in performance again.

Detection Method	**Recall**	**Correctly Labelled**	**Test Features**
Dependency Tree	51%	21738	42716
Word Distribution	10%	20	206
Word Embedding	39%	228	587

Table 6.3: Evaluation metrics on the Japanese dataset at feature-level

6.4.2 Testing at Document-level

German

In this experiment, the classifiers were trained on the English-French dataset. To measure the recall, each method performed feature extraction on the machine-translated German documents, with each method extracting a different number of feature vector from the document. For each method, it's respective trained classifier predicted the class of each feature vector, and a majority vote was carried out to finally classify the document.

Table 6.4 shows the dependency tree method has the best performance in detecting machine-translated documents from German to English, with a recall of 56%. In addition, it is the only method for which performance which remains consistent for document-level and feature-level. The word distribution method has the second best performance at 36 %. Finally, the word embedding method has the worst performance, with a recall of 34%. Additionally, in comparison to the recall of 46% at feature-level, the word embedding method has a drop in performance of 12% on the same dataset at a higher level.

Detection Method	**Recall**	**Correctly Labelled**	**Test Chapters**
Dependency Tree	56%	131	245
Word Distribution	36%	89	245
Word Embedding	34%	85	245

Table 6.4: Evaluation metrics on the German dataset at document-level

Japanese

The experiment setup is the same as the experiment for machine-translated German documents, with the change in the dataset to machine-translated Japanese documents.

Table 6.5 shows the dependency tree method has the best performance in detecting machine-translated documents from Japanese to English, with a recall of 54%. This is similar to the performance in detecting machine-translate German documents. The word distribution has negligible performance in detecting machine-translated documents. Surprisingly, the word embedding method has better performance in detecting the Japanese dataset than the German dataset at document-level, and it has better performance at detecting Japanese dataset at document-level than at feature-level. This suggests the performance of some methods such as the word distribution method is very sensitive to linguistic distance.

Detection Method	**Recall**	**Correctly Labelled**	**Test Chapters**
Dependency Tree	54%	112	206
Word Distribution	10%	20	206
Word Embedding	45%	92	206

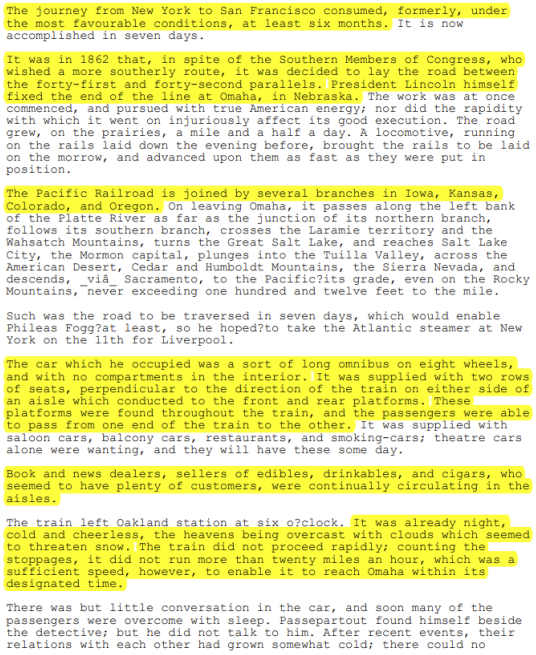
Table 6.5: Evaluation metrics on the Japanese dataset at document-level

6.5 Web-app

This section describes the features of the implemented web-app, and improvements that could be made.

Figure 6.1 is the highlighted text produced from the web-app with the selected method as dependency tree, and the input document as Chapter 26 from "Around the World in Eighty Days" [54]. The highlighted segments are clearly identifiable, and the text is readable.

However, the produced text could be better as there are symbols which shouldn't be appearing in the middle of the words. For example, the sentence "The train left Oakland station at six o? clock" should not have a question mark, but an apostrophe instead.



The journey from New York to San Francisco consumed, formerly, under the most favourable conditions, at least six months. It is now accomplished in seven days.

It was in 1862 that, in spite of the Southern Members of Congress, who wished a more southerly route, it was decided to lay the road between the forty-first and forty-second parallels. President Lincoln himself fixed the end of the line at Omaha, in Nebraska. The work was at once commenced, and pursued with true American energy; nor did the rapidity with which it went on injuriously affect its good execution. The road grew, on the prairies, a mile and a half a day. A locomotive, running on the rails laid down the evening before, brought the rails to be laid on the morrow, and advanced upon them as fast as they were put in position.

The Pacific Railroad is joined by several branches in Iowa, Kansas, Colorado, and Oregon. On leaving Omaha, it passes along the left bank of the Platte River as far as the junction of its northern branch, follows its southern branch, crosses the Laramie territory and the Wahsatch Mountains, turns the Great Salt Lake, and reaches Salt Lake City, the Mormon capital, plunges into the Tuilla Valley, across the American Desert, Cedar and Humboldt Mountains, the Sierra Nevada, and descends, _viâ_ Sacramento, to the Pacific?its grade, even on the Rocky Mountains, never exceeding one hundred and twelve feet to the mile.

Such was the road to be traversed in seven days, which would enable Phileas Fogg?at least, so he hoped?to take the Atlantic steamer at New York on the 11th for Liverpool.

The car which he occupied was a sort of long omnibus on eight wheels, and with no compartments in the interior. It was supplied with two rows of seats, perpendicular to the direction of the train on either side of an aisle which conducted to the front and rear platforms. These platforms were found throughout the train, and the passengers were able to pass from one end of the train to the other. It was supplied with saloon cars, balcony cars, restaurants, and smoking-cars; theatre cars alone were wanting, and they will have these some day.

Book and news dealers, sellers of edibles, drinkables, and cigars, who seemed to have plenty of customers, were continually circulating in the aisles.

The train left Oakland station at six o?clock. It was already night, cold and cheerless, the heavens being overcast with clouds which seemed to threaten snow. The train did not proceed rapidly; counting the stoppages, it did not run more than twenty miles an hour, which was a sufficient speed, however, to enable it to reach Omaha within its designated time.

There was but little conversation in the car, and soon many of the passengers were overcome with sleep. Passepartout found himself beside the detective; but he did not talk to him. After recent events, their relations with each other had grown somewhat cold; there could no

Figure 6.1: Highlighted text from "Around the World in Eighty Days" (which has been machine-translated)

Figure 6.2 is the main page of the web-app, where users can upload a txt file, and

select a method to perform textual analysis. Figure 6.3 is what the page looks like after the submit button has been pressed. A progress bar is displayed to indicate the progress made in classifying the document. Once textual analysis has been performed, the result and confidence level is displayed, and a highlighted text file is downloaded.

The main page is simple and easy to follow, and the progress bar is indicative of the actual progress made. However, how the confidence level is calculated could be improved. Currently, the confidence level is the percentage of features which were classified as machine-translated from the total features extracted. Furthermore, the main page should allow batch file uploads. Currently, only one file is allowed during upload. It would be much more convenient since markers would have hundreds of courseworks to sift through.

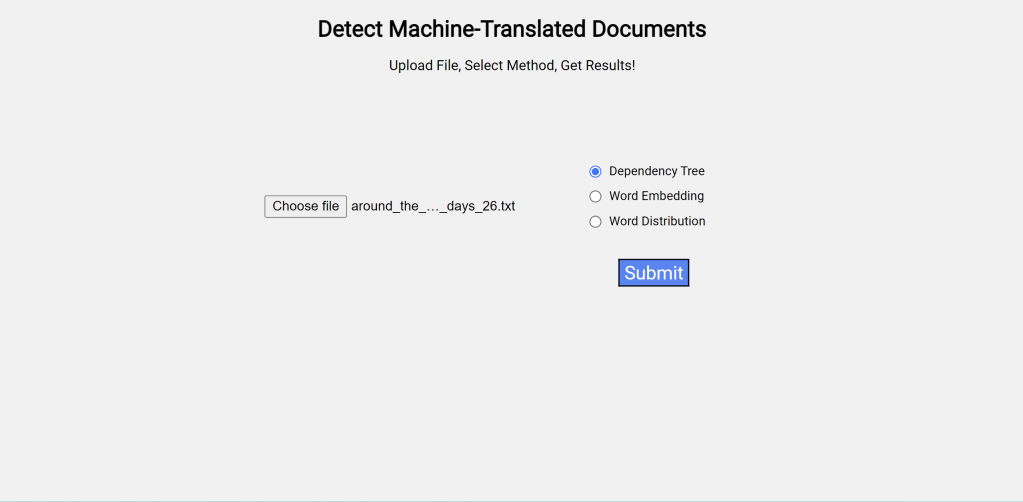


Figure 6.2: Main page

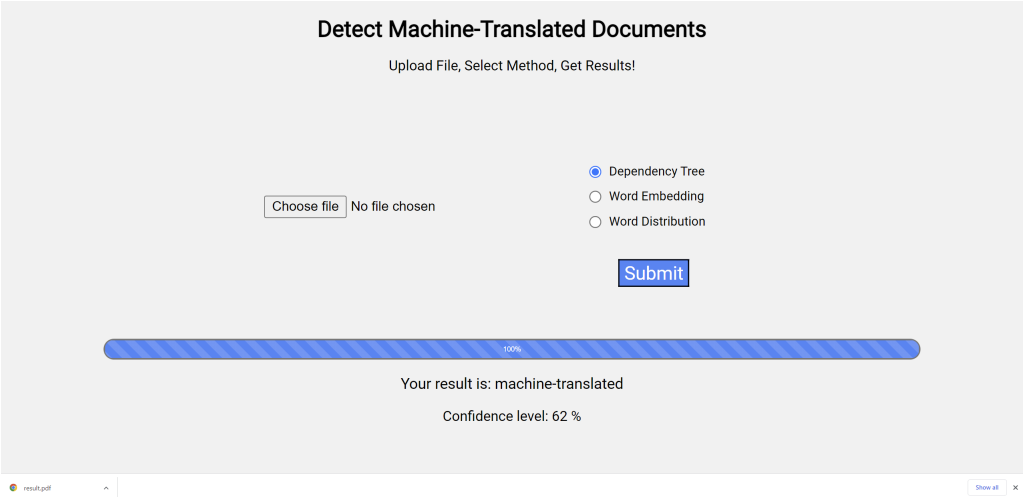


Figure 6.3: Main page after classification

6.6 Discussion

From our investigation, the highest performing method to detect machine-translation in the same language-pair as the training dataset at feature-level was the word embedding method. However, this method does not translate well across different language-pairs. It also does not translate well when used at document-level.

The most versatile method was the dependency tree method. It translated well across all language-pairs, and across from feature-level to document-level.

The word distribution method performed well in the first experiment. However, its performance significantly dropped to the point where it was negligible when the language-pair was changed from English-French to German, and then to Japanese.

The back-translation method was only performed for the first experiment because the speed of feature extraction was significantly limited. In the same amount of time the dependency tree method took to perform feature extraction on 480828 features from the English-French dataset, the back-translation method could only parse 4604 features. This led us to abandon the method in further experiments, as the performance initially was not good to begin with.

Chapter 7

Conclusion

Overall, we have met the objectives set out at the start of this project.

1. **Investigate which features of a text are relevant in detecting machine-translation**
   We have implemented past methods which use back-translation, dependency tree, word distribution, and word embedding. We did not implement the phrase salad method [25], as it relied on identifying phrase salads which are not as prevalent in NMT systems as it was in SMT systems.

2. **Evaluate which method performs best through a series of experiments**
   We have determined accuracy, recall, and f1 score to be the suitable evaluation metrics. In addition, we found the dependency tree to be the most consistent and reliable method using these evaluation metrics in our evaluation.

3. **Extract relevant information from the input text**
   We developed a web-app, which classifies whether an input document is `machine translated` or `human written`. In the case the document is machine-translated, the document is highlighted to identify the key segments of text which was machine-translated.

7.1 Future Work

There is still great scope for improvement. This section discusses the direction and steps that would be taken to improve the project.

7.1.1 Test performance in training a different dataset

Our classifier was trained on the English-French dataset. In our experiments, we found that machine-translated documents are easier to detect in the same language-pair as the trained model. We should have further tests to see if it holds for different language pairs than English-French. We could not train the classifier on the English-Japanese dataset, because there are not enough Japanese eBooks listed on Project Gutenberg.

7.1.2 Test performance in different genre domain

Our dataset had a range of different genres of books. We could perform an experiment to find out whether certain genres of books were detected better than others. Additionally, an experiment could be conducted to find whether a classifier trained on a certain genre of books would be better in detecting those genres. If this experiment supports this hypothesis, we could improve the current software by implementing a genre detector first, and then assigning the task for that.

7.1.3 Better Method for document classification

The performance of the classifier at document-level could definitely be improved. Our best classifier, which is the dependency tree method, achieves 56% and 50% for German and Japanese documents.

The current method of document classification is a simple approach. From our findings, we could several trained classifiers trained in different languages. A document it is labelled as `machine-translated` if one classifier spots this as `machine-translated`. If serialised, this approach could take a long time. However, if we are able to parallelise this process, it would be roughly equal to the same time as one classifier.

7.1.4 Evaluate performance of classifier

Finding which feature is relevant in detecting machine-translation is not the entire task. The neural network architecture which is then used should also be considered. In future work, investigating which classifier performs best would help in the problem of detecting machine-translated documents. Then in theory, the combination of the relevant feature. However, this might not always be the case.

7.1.5 Classification to Regression

In our experiments, when classifying a document, we assumed that a document is either machine-translated or human-written. In a realistic scenario, it is more likely the case that a document is partially machine-translated. Therefore, our approach of majority voting would be flawed. For example, assuming the dependency tree method was 100% accurate at sentence-level, if 40% of the sentences in a document was machine-translated, the document would be classified as `human-written`. Therefore, the task for detecting machine-translation in document-level should be a regression task instead.

7.2 Findings

To conclude, we summarise the main findings of this project:

- The highest performance for any method is achieved when the trained classifier has the same language-pair as the input document.

- The correlation between the linguistic distance of trained classifier and input document, and performance of the detection method is dependent on the method. For dependency tree method the correlation is weak, but for the word distribution and word embedding method it is very strong.
- The further the linguistic distance of the trained classifier to the input document, the harder it is to detect machine-translation. This is shown by the drop in performance across all methods in table 6.1, table 6.2, and table 6.3.
- The dependency tree method is the most consistent and reliable method with an average recall of 54% across different language-pair and at document level.
- The word embedding method has the highest performance with an accuracy of 73% when using the trained classifier has the same language-pair as the input document, and it is evaluated at feature-level.

Bibliography

[1] Läubli S, Sennrich R, Volk M. Has Machine Translation Achieved Human Parity? A Case for Document-level Evaluation. *In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4791–4796, 2018. Available at https://aclanthology.org/D18-1512.

[2] National Research Council. *Language and Machines: Computers in Translation and Linguistics*. Washington, DC: The National Academies Press, 1966. Available at https://doi.org/10.17226/9547.

[3] Okpor MD. Machine Translation Approaches: Issues and Challenges. *International Journal of Computer Science Issues*, 11:159–165, 2014.

[4] Nagao M. A Framework of a Mechanical Translation between Japanese and English by Analogy Principle. *In Proceedings of the International NATO Symposium on Artificial and Human Intelligence*, pages 173–180, 1984. Available at https://dl.acm.org/doi/10.5555/2927.2938.

[5] Lopez A. Statistical Machine Translation. *ACM Computing Surveys*, 40:1–49, 2008. Available at https://doi.org/10.1145/1380584.1380586.

[6] Knight K, Koehn P. What's New in Statistical Machine Translation. *Tutorial at HLT-NAACL*, pages 1–89, 2003. Available at https://people.csail.mit.edu/koehn/publications.

[7] Brown PF, Cocke J, Della Pietra SA, Della Pietra VJ, Jelinek F, et al. A Statistical Approach to Machine Translation. *Computational Linguistics*, 16(2):79–85, 1990. Available at https://aclanthology.org/J90-2002.

[8] Knight K. Automating Knowledge Acquisition for Machine Translation. *AI Magazine*, 18(4):1–16, 1997. Available at https://doi.org/10.1609/aimag.v18i4.1323.

[9] Brownlee J. *A Gentle Introduction to Neural Machine Translation*. Available at https://machinelearningmastery.com/introduction-neural-machine-translation.

[10] Wu Y, Schuster M, Chen Z, Le QV, Norouzi M, et al. Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. *CoRR*, pages 1–23, 2016. Available at https://arxiv.org/abs/1609.08144.

[11] Brownlee J. *What Is Natural Language Processing?* Available at https://machinelearningmastery.com/natural-language-processing.

[12] Manning CD, Raghavan P, Schuetze H. *Introduction to Information Retrieval*. Cambridge University Press, 2008. Available at https://nlp.stanford.edu/IR-book/information-retrieval-book.html.

[13] Schmid H. Part-of-Speech Tagging with Neural Networks. *In COLING 1994 Volume 1: The 15th International Conference on Computational Linguistics*, pages 1–5, 1994. Available at https://aclanthology.org/C94-1027.

[14] NLTK. *NLTK Documentation*. Available at https://www.nltk.org.

[15] SpaCy. *SpaCy Documentation*. Available at https://spacy.io.

[16] Jurafsky D, Martin JH. *Dependency Parsing. In: Speech and Language Processing*. Pearson Prentice Hall, 2000. Available at https://web.stanford.edu/~jurafsky/slp3.

[17] SpaCy. *Visualizing the dependency parse*. Available at https://spacy.io/usage/visualizers.

[18] Brownlee J. *What Are Word Embeddings for Text?* Available at https://machinelearningmastery.com/what-are-word-embeddings.

[19] Mikolov T, Chen K, Corrado G, Dean J. Efficient Estimation of Word Representations in Vector Space. *ICLR Workshop Track*, pages 1–12, 2013. Available at https://arxiv.org/abs/1301.3781.

[20] Rong X. Word2vec Parameter Learning Explained. pages 1–21, 2014. Available at https://arxiv.org/abs/1411.2738.

[21] Pennington J, Socher R, Manning C. GloVe: Global Vectors for Word Representation. *In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1532–1543, 2014. Available at https://aclanthology.org/D14-1162.

[22] Cortes C, Vapnik V. Support-Vector Networks. *Machine Learning Research*, 20:273–297, 1995. Available at https://doi.org/10.1007/BF00994018.

[23] Stewart M. *Simple Introduction to Convolutional Neural Networks*. Available at https://towardsdatascience.com/simple-introduction-to-convolutional-neural-networks-cdf8d3077bac.

[24] Newatia R. *How to implement CNN for NLP tasks like Sentence Classification*. Available at https://medium.com/saarthi-ai/sentence-classification-using-convolutional-neural-networks-ddad72c7048c.

[25] Arase Y, Zhou M. Machine Translation Detection from Monolingual Web-Text. *In Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, 1:1597–1607, 2013. Available at https://aclanthology.org/P13-1157.

[26] Bansal M, Quirk C, Moore R. Gappy Phrasal Alignment By Agreement. *In Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1308–1317, 2011. Available at https://aclanthology.org/P11-1131.

[27] Brislin RW. Back-Translation for Cross-Cultural Research. *Journal of Cross-Cultural Psychology*, pages 185–216, 1970. Available at https://doi.org/10.1177/135910457000100301.

[28] Vanmassenhove E, Shterionov D, Way A. Lost in Translation: Loss and Decay of Linguistic Richness in Machine Translation. *In Proceedings of Machine Translation Summit XVII: Research Track*, pages 222–232, 2019. Avaliable at https://aclanthology.org/W19-6622.

[29] Nguyen-Son H-Q, Thao TP, Hidano S, Kiyomoto S. Detecting Machine-Translated Text using Back Translation. *In Proceedings of the 12th International Conference on Natural Language Generation*, pages 189–197, 2019. Available at https://arxiv.org/abs/1910.06558.

[30] Koehn P. Europarl: A Parallel Corpus for Statistical Machine Translation. *In Proceedings of Machine Translation Summit X: Papers*, pages 79–86, 2005. Available at https://aclanthology.org/2005.mtsummit-papers.11.

[31] Tatman R. *Evaluating Text Output in NLP: BLEU at your own risk*. Available at https://towardsdatascience.com/evaluating-text-output-in-nlp-bleu-at-your-own-risk-e8609665a213.

[32] Zipf GK. *The Psycho-Biology of Language: An Introduction to Dynamic Philology*. London: Routledge, 1935.

[33] Solity J, Vousden J. *Real books vs reading schemes: a new perspective from instructional psychology*. Educational Psychology, 2009. Available at https://doi.org/10.1080/01443410903103657.

[34] Nguyen-Son H-Q, Tieu N-D, Nguyen H, Yamagishi J, Echizen I. Identifying Computer-Generated Text Using Statistical Analysis. *Asia-Pacific Signal and Information Processing Association Annual Summit and Conference*, pages 1–8, 2017. Available at https://ieeexplore.ieee.org/document/8282270.

[35] *Project Gutenberg*. Available at https://gutenberg.org.

[36] Chae J, Nenkova A. Predicting the Fluency of Text with Shallow Structural Features: Case Studies of Machine Translation and Human-Written Text. *In Proceedings of the 12th Conference of the European Chapter of the ACL*, pages 139–147, 2009. Available at https://aclanthology.org/E09-1017.

[37] Charniak E, Johnson M. Coarse-to-Fine n-Best Parsing and MaxEnt Discriminative Reranking. *In Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, pages 173–180, 2005. Available at https://aclanthology.org/P05-1022.

[38] Li Y, Wang R, Zhai H. A Machine Learning Method to Distinguish Machine Translation from Human Translation. *In Proceedings of the 29th Pacific Asia Conference on Language, Information and Computation: Posters*, pages 354–360, 2015. Available at https://aclanthology.org/Y15-2041.

[39] Marneffe M-CD, MacCartney B, Manning CD. Generating Typed Dependency Parses from Phrase Structure Parses. *In Proceedings of the Fifth International Conference on Language Resources and Evaluation*, pages 1–6, 2006. Available at https://aclanthology.org/L06-1260.

[40] Glottopedia. *Coherence Definition*. Available at http://www.glottopedia.org/index.php/Coherence.

[41] Nguyen-Son H-Q, Thao TP, Hidano S, Kiyomoto S. Detecting Machine-Translated Paragraphs by Matching Similar Words. *ArXiv*, pages 1–12, 2019. Available at https://arxiv.org/abs/1904.10641.

[42] Manning C, Surdeanu M, Bauer J, Finkel J, Bethard S, et al. The Stanford CoreNLP Natural Language Processing Toolkit. *In Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60, 2014. Available at https://aclanthology.org/P14-5010.

[43] *Hugging Face*. Available at https://huggingface.co.

[44] *Scikit-learn Documentation*. Available at https://scikit-learn.org.

[45] *Python Documentation*. Available at https://www.python.org.

[46] *C++ Documentation*. Available at https://isocpp.org/std/the-standard.

[47] *Java Documentation*. Available at https://docs.oracle.com/en/java.

[48] *Django Documentation*. Available at https://docs.djangoproject.com.

[49] *Flask Documentation*. Available at https://flask.palletsprojects.com.

[50] *Googletrans Documentation*. Available at https://pypi.org/project/googletrans.

[51] Ng N, Yee K, Baevski A, Ott M, Auli M, et al. Facebook FAIR's WMT19 News Translation Task Submission. *In Proceedings of the Fourth Conference on Machine Translation*, 2:314–319, 2019. Available at https://aclanthology.org/W19-5333.

[52] *Google Translate*. Available at https://translate.google.co.uk.

[53] Chiswick BR, Miller PW. Linguistic Distance: A Quantitative Measure of the Distance Between English and Other Languages. *Journal of Multilingual and Multicultural Development*, 26(1):1–11, 2005. Available at https://doi.org/10.1080/14790710508668395.

[54] Verne J, Holeinone P. *Around the World in Eighty Days*. France:Le Temps, 1872.

[55] *SpaCy POS Tags*. Available at https://machinelearningknowledge.ai/tutorial-on-spacy-part-of-speech-pos-tagging.

[56] *Penn Treebank Project POS Tags*. Available at https://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html.

[57] *Universal Dependencies*. Available at https://universaldependencies.org/u/dep/all.html.

Appendix A

SpaCy POS Tag Definitions

POS Tag	Description	Examples
ADJ	Adjective	old, yellow, first
ADP	Adposition	in, to, during
ADV	Adverb	very, tomorrow, down, there
AUX	Auxiliary	is (doing), has (done), will (do)
CONJ	Conjunction	and, or, but
DET	Determiner	a, an, the
INTJ	Interjection	psst, ouch, bravo, hello
NOUN	Noun	girl, cat, tree
NUM	Numeral	1, 2017, one, billion, IV
PART	Particle	’s, not
PRON	Pronoun	I, you, he, she
PROPN	Proper Noun	Steve, Tokyo
PUNCT	Punctuation	. ! ?
SCONJ	Subordinating Conjunction	if, while, that
SYM	Symbol	$, %, +
VERB	Verb	eat, ate, eating
X	Other	asdfgh, irhngbf

Table A.1: SpaCy POS Tags List [55]

Appendix B

NLTK POS Tag Definitions

POS Tag	Description
CC	Coordination Conjunction
CD	Cardinal Number
DT	Determiner
EX	Existential There
FW	Foreign Word
IN	Preposition
JJ	Adjective
JJR	Adjective, comparative
JJS	Adjective, superlative
LS	List Item Marker
MD	Modal
NN	Noun, singular or mass
NNP	Proper Noun, singular
NNPS	Proper Noun, plural
PDT	Predeterminer
POS	Possessive ending
PRP	Personal Pronoun
PRP$	Possessive Pronoun
RB	Adverb
RBR	Adverb, comparative
RBS	Adverb, superlative
RP	Particle
SYM	Symbol
TO	*to*
UH	Interjection
VB	Verb, base form
VBD	Verb, past tense

Table B.1: NLTK POS Tags List [56]

POS Tag	Description
VBG	Verb, ground or present participle
VBN	Verb, past participle
VBP	Verb, non-3rd person singular present
VBZ	Verb, 3rd person singular present
WDT	Wh-determiner
WP	Wh-pronoun
WP$	Possessive wh-pronoun
WRB	Wh-adverb

Table B.2: Continued NLTK POS Tags List [56]

Appendix C

Dependency Tag Definitions

For brevity, variations of the same tag have been grouped together. For example, `advmod:emph` and `advmod:lmod` are both grouped as `advmod`.

Dependency Tag	**Description**
acl	clausal modifier of noun (adnominal clause)
advmod	adverbial modifier
amod	adjectival modifier
appos	appositional modifier
aux	auxiliary
case	case marking
cc	coordinating conjunction
ccomp	clausal complement
clf	classifier
compound	compound
conj	conjunct
cop	copula
csubj	clausal subject
dep	unspecified dependency
det	determiner
discourse	discourse element
dislocated	dislocated elements
expl	expletive
fixed	fixed multiword expression
flat	flat multiword expression
goeswith	goes with
iobj	indirect object
list	list
mark	marker

Table C.1: Universal Dependencies Tag List [57]

Dependency Tag	Description
nmod	nominal modifier
nsubj	nominal subject
nummod	numeric modifier
obj	object
obl	oblique nominal
orphan	orphan
parataxis	parataxis
punct	punctuation
reparandum	overridden disfluency
root	root
vocative	vocative
xcomp	open clausal complement

Table C.2: Continued Universal Dependencies Tag List [57]