

IMPERIAL COLLEGE LONDON

DEPARTMENT OF COMPUTING

Lost in translation: fidelity-focused machine translation evaluation

Author:
Julian Chow

Supervisor:
Dr. Lucia Specia

Submitted in partial fulfillment of the requirements for the BEng degree in Computing of Imperial College London

June 2019

Abstract

Translation is a universal problem which has applications in many contexts. Source texts are frequently translated from one language to another to aid communication across the world. Machine translations are one of the most popular applications of natural language processing.

Evaluating the quality of a translation, particularly machine translation, is traditionally done by a human annotator, which remains the “gold standard” of the field. This is typically based off factors of adequacy and fluency.

Automated metrics to evaluate translation quality exist and are generally based off correlation with the gold standard human annotation. Traditionally, these metrics involve measures of string similarity and pattern matching. These metrics are much cheaper and faster than human evaluation.

This paper presents a new automated metric for translation evaluation, moving away from string similarity and focusing instead on semantic similarity between words. This approach is much more flexible and allows the message and tone of the original text to be better preserved. This is based on the word embeddings of neural word distributional models and distance metrics based on vector spaces. Building upon the Word Mover’s distance as a measure within the semantic space, the metric introduces a fragmentation penalty and a missing word penalty to account for the neglect of fluency in the default WMD bag-of-words approach.

This word order extension is proven to perform better than standard WMD, with promising results against other state-of-the-art metrics, including far better performance in comparison with traditional string matching metrics.

Finally, it is hoped insights provided through this metric provide a building block for the future of machine translation evaluation. In particular, the concept of using embeddings hopes to make possible the use of cross-lingual embeddings in the future to allow direct evaluation of a source text and its machine translation, without the need for a reference human translation as currently is the case.

Acknowledgements

I would like to thank my supervisor, Lucia Specia, for all her help throughout this project. I would also like to thank Pranava Madhyastha for his useful guidance, as well as Francesca Toni for her aid as a second marker.

I would also like to thank my friends that have helped me with advice throughout this project and my three years at Imperial.

Most importantly I would like to thank my family for their continued support, in particular my father, who sadly passed away in February 2019. I think about your company often and miss you greatly.

Lastly, I want to thank the people of Hong Kong for standing up for our human rights and continuing the fight for freedom. Not being able to join them at this time is a big regret of mine.

Contents

I	Introduction	1
1	Introduction	2
1.1	Motivations	2
1.2	Objectives	2
2	Background	4
2.1	Human evaluation	4
2.1.1	Rating based	4
2.1.2	Ranking based	6
2.1.3	Post-editing judgment	6
2.2	Automatic evaluation	7
2.2.1	Edit distance	7
2.2.2	BLEU	8
2.2.3	NIST	10
2.2.4	ROUGE	11
2.2.5	METEOR	13
2.3	Distributional Semantics	15
2.3.1	Count-based methods	15
2.3.2	Predictive methods	17
2.3.3	Word2vec	19
2.3.4	fastText	21
2.4	Distance metrics	21
2.4.1	Cosine similarity	21
2.4.2	Euclidean distance	22
2.4.3	Earth Mover's Distance	22
2.4.4	Word Mover's Distance	23
II	Method Overview and Implementation	26
3	Implementation	27
3.1	Reference translations	27
3.1.1	Datasets	28
3.1.2	Preprocessing	30
3.2	Word Mover's Distance	31
3.3	Word Embeddings	32
3.4	Tweaking WMD	33
3.4.1	Distance measure	33
3.4.2	Missing words	34
3.4.3	Stop words	36

3.5	Baseline WMD results	37
3.6	Word order penalties	39
3.7	Fragmentation penalties	43
3.7.1	Additive fragmentation penalty	45
3.7.2	Multiplicative fragmentation penalty	46
3.8	Tackling anomalies	49
3.8.1	Missing word penalty	49
3.8.2	Number vector	49
3.9	Finalising metric	50
3.10	Alternative approaches not used	51
III	Conclusions	52
4	Results	53
4.1	Performance against other metrics	53
4.2	Analysis	54
4.3	Anomalous sentences	60
5	Evaluation	64
5.1	Metric performance	64
5.2	Testing	65
5.3	Implementation	66
5.4	Usage	66
5.5	Qualitative feedback	67
6	Conclusions & Future Work	69
6.1	Lessons learnt	69
6.2	Future Work	70

Part I

Introduction

Chapter 1

Introduction

1.1 Motivations

Translation is the idea of communicating the meaning of a text from a source language to a different target language. Translation of texts between different languages has been necessary throughout history to exchange ideas and communication between groups of different cultures, paving the way for the globalisation of today. Quality translation plays a pivotal role in international affairs, from the global scale of politics to the local corner shops welcoming tourists. Having accurate and concise translations allows people to understand messages from all different types of sources, especially when they are not speakers of the source language and would not be exposed to the context otherwise.

Traditional human translations are done by an expert translator, who has knowledge and understanding of the source and target language both in syntax and in cultural nuance. However, manual human translations are time-consuming and expensive, meaning they cannot be conducted at great scale. Automated machine translations have been developed with the growth of computing to mitigate the labour-intense work of translations. In contrast to traditional human translations, machine translations do not have the benefit of the complex cognitive abilities a professional translator has, instead having to rely on programmed understanding to decompose a text into its fundamental meaning. It also has the challenge of piecing the text into a fluent expression in the target language. As a result, machine translations under current technology are often imperfect, but provide a good approximation of the original text both in meaning and in style.

Evaluating the quality of a machine translation is paramount to the continual improvement of machine translation systems, as they provide a strong indicator of how closely the system's outputs correlate with their human counterparts. When assessing the performance of these translation systems, there is no explicit criteria as to what constitutes as a good translation, as this is a very subjective area. Different aspects can be considered at different priorities, such as how fluent a translation is, or how much meaning is preserved. As a result, there are a large number of different methods to evaluate machine translation systems. Human evaluations tend to evaluate the translation as a whole, judging it on its perceived quality, whereas automated metrics often compare the translated output with a pre-created human translation. This is often measuring string similarity between the two, made possible because both of these texts are in the same language. Automatic evaluation metrics allow for a much cheaper and faster rating than human evaluation, but may be less reliable without the focus of a human judge.

1.2 Objectives

However, the approach of the existing automated metrics ignores the source text and its contained meaning. This makes it harder to evaluate if there has been any deviation from the semantics of

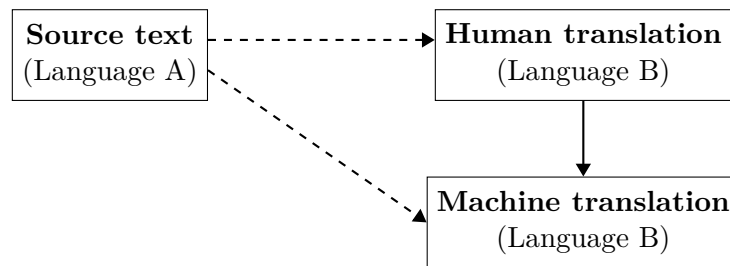


Figure 1.1: Flow of existing automatic evaluation metrics.

the original text in either the machine translation or the human translation. Given the difficulty of translating a text even for a human, it is not unlikely that a reference translation used by the automated metrics omits certain contextual elements that are rich in human language. Figure 1.1 illustrates this: dotted lines represent where there is a translation between texts, while the solid line represents where the actual comparison takes place. It can be seen that the actual comparison only takes place between the two translations, rather than with the source text.

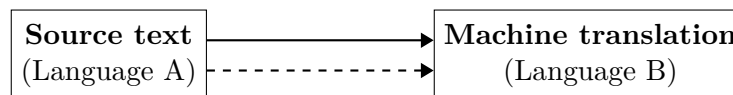


Figure 1.2: The objective flow of this project's evaluation metric.

The objective of this project is to create an evaluation metric that focuses on semantic similarity rather than string similarity. The idea is to implement semantics of a text using neural word distributional models, then evaluate translations using appropriate distance metrics. Directly using word meaning in assessing the quality of a translation can better preserve the message and tone of the original text. This approach can also eliminate the need for an intermediary human reference translation in evaluations, as the two texts can be directly compared despite being in two different languages. This is illustrated in Figure 1.2.

Chapter 2

Background

2.1 Human evaluation

In translation evaluation, humans are ultimately the ones interpreting translated texts. Naturally, human evaluation has become the gold standard for assessing the accuracy and efficiency of a machine translation. While manual evaluation may provide the most accurate and reliable ratings for translation systems, it is both time-consuming and expensive to corral a group of qualified and unbiased people to partake in studies, making it difficult to use widely. In addition, there can be many correct translations to a single text, making it hard to settle on a universally faultless translation for each source. Consistency across multiple judges, and even the same judge, can be difficult to achieve, even among expert translators. However, most human methods of evaluating a translation aim to quantify a translation’s quality, reducing ambiguity with a set of specific criteria. Such inter-annotator and intra-annotator disagreement can also be effectively mitigated with a large sample size [18], making it a very powerful tool in evaluating the capacity of a machine translation.

2.1.1 Rating based

One of the first instances of human evaluation [12] measured intelligibility and informativeness on a scale from 1 to 9 to find the translation’s standard. Characterising intelligibility as a wording that reads as ordinary text with no stylistic errors and informativeness as a measure of how close the meaning is to the original text’s intention, the simple scale produced results which closely correlated the two criteria, along with that of reading time. While there is the argument to make that the intelligibility scale can be multidimensional in that the high end ratings look at more nuanced matters of word choice whereas the low end ratings focus on rudimentary syntax order, translations were found to naturally tend along this single line. This method formulated by ALPAC paved the way for other rating-based human assessment systems, targeting a range of other factors.

The ARPA machine translation evaluation metrics [64] targeted adequacy, fluency and comprehension as aspects of a successful translated text. To cope with the smaller amount of human talent available, the researchers created targeted metrics for judges to make decisions on. The adequacy aspect assessed the degree in which fragments of information from a professional reference translation could be found in a given machine translation. The fluency assessment was done by evaluating each sentence, determining whether it was well-formed and linguistically appropriate. The comprehension evaluation scored ratings on six different questions to determine how successful information transmittal was in the translation. Normalising the results for each of the factors to be between 0 and 1, the team were able to create a sense of scale for different machine translation systems. These three metrics have been used as the basis of many human evaluation studies [26] following this.

Nevertheless, several challenges can still arise from methodological factors both in this research and in going forward with any human evaluation. Human performance can differ based on experience,

	Intelligibility	Informativeness
9	Perfectly clear and intelligible. Reads like ordinary text; has no stylistic infelicities.	Extremely informative. Makes “all the difference in the world” in comprehending the meaning intended. (A rating of 9 should always be assigned when the original completely changes or reverses the meaning conveyed by the translation.)
8	Perfectly or almost clear and intelligible but contains minor grammatical or stylistic infelicities and/or mildly unusual word usage that could, nevertheless, be easily “corrected.”	Very informative. Contributes a great deal to the clarification of the meaning intended. By correcting sentence structure, words, and phrases, it makes a great change in the reader’s impression of the meaning intended, although not so much as to change or reverse the meaning completely.
7	Generally clear and intelligible, but style and word choice and/or syntactical arrangement are somewhat poorer than in category 8.	Between 6 and 8.
6	The general idea is almost immediately intelligible, but full comprehension is distinctly interfered with by poor style, poor word choice, alternative expressions, untranslated words, and incorrect grammatical arrangements. Postediting could leave this in nearly acceptable form.	Clearly informative. Adds considerable information about the sentence structure and individual words, putting the reader “on the right track” as to the meaning intended.
5	The general idea is intelligible only after considerable study, but after this study one is fairly confident that he understands. Poor word choice, grotesque syntactic arrangement, untranslated words, and similar phenomena are present but constitute mainly “noise” through which the main idea is still perceptible.	Between 4 and 6.
4	Masquerades as an intelligible sentence, but actually it is more unintelligible than intelligible. Nevertheless, the idea can still be vaguely apprehended. Word choice, syntactic arrangement, and/or alternative expressions are generally bizarre, and there may be critical words untranslated.	In contrast to 3, adds a certain amount of information about the sentence structure and syntactical relationships. It may also correct minor misapprehensions about the general meaning of the sentence or the meaning of individual words.
3	Generally unintelligible; it tends to read like nonsense, but with a considerable amount of reflection and study, one can at least hypothesize the idea intended by the sentence.	By correcting one or two possibly critical meanings, chiefly on the word level, it gives a slightly different “twist” to the meaning conveyed by the translation. It adds no new information about sentence structure, however.
2	Almost hopelessly unintelligible even after reflection and study. Nevertheless it does not seem completely nonsensical.	No really new meaning is added by the original, either at the word level or the grammatical level, but the reader is somewhat more confident that he apprehends the meaning intended.
1	Hopelessly unintelligible. It appears that no amount of study and reflection would reveal the thought of the sentence.	Not informative at all; no new meaning is added nor is the reader’s confidence in his understanding increased or enhanced.
0		The original contains, if anything, less information than the translation. The translator has added certain meanings, apparently to make the passage more understandable.

Table 2.1: Scale of intelligibility and informativeness from the ALPAC study of 1966 [12].

creating a variance in judgment. Some people may be more familiar with certain styles of writing or types of vocabulary, giving them a different perspective from that of another reader. The order of reading different texts can also create unconscious biases in evaluation [64]. Annotators also disagree on which parts of the sentence are most important when assigning ratings, with conflict on which phrases are most vital. The length of the sentence can similarly cause difficulty. If it is too long it can be muddled and hard to classify correctly, whereas if it is too short it can fall between boundaries on the adequacy front [18]. Even more so, fatigue of the reader can contribute to different ratings at different times. Further studies [30] have suggested that even with an explicit description for each level of rating on the scale, annotators can struggle to maintain a consistent standard throughout, preferring a ranking based system instead.

2.1.2 Ranking based

In a ranking based system, judges are given a set of translations and asked to rank them from best to worst. This task replaces the arbitrary scales of the rating system with a series of relative judgments, removing the difficulty in assigning a numerical value for each criteria to each translation [62]. However, comparing two nearly identical translations can be confusing, as can be having to compare one error’s severity with another. Judges largely have to decide for themselves which errors have the greater impact on the translation’s quality.

Introducing multiple evaluators in this system can be difficult when combining conflicting annotations; in a ranking system, scores cannot be averaged as they would in a rating system as this would ignore the pairwise rankings for each given annotation. This can cause a chunk of ratings to be invalidated in tasks like tuning an automatic evaluation metric, which uses rank consistency in calculation [18].

Nevertheless, collected rankings still can be used assign a score to translation systems. One option is to randomly give judges translations of five systems to rank from 1 to 5, allowing for ties [9]. With the following formula, the pairwise rankings can be used to rate how frequently a system A was rated to be better than another system B :

$$\frac{\text{\#A rated higher than B}}{\text{\#comparisons between A and B} - \text{\#A tied with B}}$$

2.1.3 Post-editing judgment

An alternative approach to directly obtaining an absolute or relative judgment of translation quality, post-editing methods measure the amount of editing required by a human translator to transform the machine translation into an acceptable semantically equivalent reference translation. The human translation error rate (HTER) [58] is based on the minimum edit distance, taking into account insertions, deletions, substitutions, and shifts to create an appropriate reference.

$$\text{HTER} = \frac{\text{\#insertions} + \text{\#deletions} + \text{\#substitutions} + \text{\#shifts}}{\text{\#words in average reference translation}}$$

As no quantitative ratings are assigned when making judgments, awkward decisions about which attributes are necessary in a good translation and how severely certain errors should be penalised are entirely avoided. Long sentences are also less of an issue as post-editors can just correct the sentence incrementally rather than having to come up with a single rating for the entire sentence. The inclusion of shifts in the edit distance also reduces any excessively heavy penalty for incorrect word order.

However, the HTER has several inherent weaknesses. All types of edits, be it insertion, deletion, substitution, or shift carry the same weight when calculating the error rate. There is also no distinction between the importance of each word; for example, a function word “*of*” would be of the same level as a content word “*peanuts*” in the sentence “*This is the flavour of peanuts*”. In addition, incorrect forms

of correct the root words are also counted as errors to substitute; an example of this is the sentence “*I’m fought for freedom*” being converted to “*I’m fighting for freedom*”, despite “*fought*” and “*fighting*” stemming from the same root word “*fight*”.

2.2 Automatic evaluation

Automatic evaluation metrics that provide a single, quantifiable score for the overall quality of a translation have become essential in the development of machine translation as they allow easy comparison of different translation systems without the high costs of human evaluators. Nevertheless, as human reference is generally taken as the ultimate metric of translation evaluation, automatic evaluation metrics are often validated by their high correlation with human judgment. Although a single score does not give a clear idea about the different dimensions of translation quality, it is an important tool in training and tuning parameters for translation models as it allows models to be iteratively improved by optimising the score. Automatic evaluation metrics differ in the way translations are compared to the reference translation; some directly compare exact lexical matches of words, while others take various strategies to compute the semantics of a translation. The basic unit of comparison can also range from single words to phrases, or even distributed representations. Being more reproducible and tunable, automated metrics can have a distinct advantage over time-consuming and expensive human evaluations in driving forward translation evaluation.

2.2.1 Edit distance

A straightforward way to measure the lexical similarity of two phrases is to calculate the minimum edit distance required to transform one to another, taking into account the insertions, deletions, and substitutions required. This works very similar to the Levenshtein distance, but calculated on words rather than characters. The word error rate [11] provides a rudimentary approach to this:

$$\text{WER} = \frac{\# \text{insertions} + \# \text{deletions} + \# \text{substitutions}}{\text{reference length}}$$

An obvious weakness to this metric and that of similar lexical measures is the issue of word order. Owing to the diversity of language, the reference sentence could be reconstructed in different ways by translations while maintaining the same meaning, which would wrongly punish a good translation. Take the following Chinese-English example:

Chinese: 由於適值農曆新年，政府會爭取在假期後得到行政會議同意，以擴大委員會的調查範圍

English (reference translation): Due to the impending Lunar New Year, the government will strive to obtain approval from the executive council after the holidays to expand the scope of the commission’s investigation.

English (translation 1): Due to the appropriate Lunar New Year, government will strive to obtain after the holidays executive council agreement, to expand commission’s investigation scope.

English (translation 2): To expand the scope of the commission’s investigation, the government will have to wait until after the impending Lunar New Year to obtain approval from the executive council.

Both sentences confer the same point of the holiday period delaying any progress to the commission’s investigation, but translation 2 would be punished far more in the standard WER metric as the word order does not match that of the reference translation, whereas translation 1 would receive a much better score as it follows the same structure, despite it being arguably less fluent.

To address the issue of word order, the position-independent word error rate [59] ignores the alignment of words in a sentence, counting the number of times identical words appear in the two

sentences. This treats the sentences as a “bag-of-words”. Words that don’t match are still substitutions and the rest are labelled as insertions or deletions depending on the length of the sentence. These measures are very quick to compute and easily done automatically; but have been suggested to not have any correlation with the accuracy of language understanding, particularly in the context of speech. If people trained features other than word recognition accuracy to optimise understanding, they would still achieve a higher accuracy of language understanding with a low word error rate [63], suggesting a full understanding needs more than high word recognition accuracy.

2.2.2 BLEU

One of the most commonly used automatic metrics for translation evaluation, BLEU (Bilingual evaluation understudy) [48] is based on the idea that the closer a machine translation is to a human translation, the better it is. Using the concept of dividing up a sentence into *n-grams*, the implementation of BLEU compares *n-grams* of the candidate translation with *n-grams* of the reference translation, counting the number of position-independent matches. Taking a single word as a unigram or *1-gram*, two consecutive words would be a *bigram* or *2-gram*, and so on.

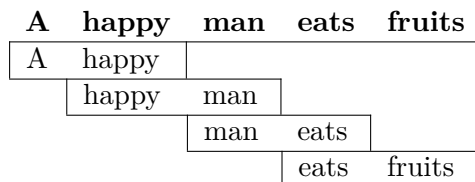


Figure 2.1: An example of the bigrams possible in the sample sentence “A happy man eats fruits.”

The more matches that are found, the better the candidate translation is. The general phenomenon found by the researchers suggested that a good translation will largely use the same words as a normal human translation, making this simple matching comparison an efficient and inexpensive tool to use.

The metric is built on the measure of precision, a measure of relevant instances from retrieved instances. The general method of calculating precision is to find a fraction of the two:

$$\text{Precision} = \frac{|(\text{relevant}) \cap (\text{retrieved})|}{|(\text{retrieved})|}$$

In the context of text translation, this would be counting the number of matches between the candidate and reference translations and dividing it by the total number retrieved from the candidate translation. An inherent weakness of this measure in translation is that the system could overproduce valid words, resulting in a high precision but nonsensical translation. Take the following example, matching on unigrams:

Candidate: An an an an.

Reference: An apple fell here.

All four words in the candidate translation match with a word “an” in the reference, which would give this candidate a perfect precision score despite it being an infeasible translation. BLEU makes a straightforward modification to this, modifying the precision calculation so that a reference word becomes exhausted and ignored in any subsequent calculations after a matching candidate word has been found for it. This modified unigram precision counts the maximum number of times a word occurs in the reference translation and caps the enumeration of each word in the reference by this number when finding relevant matches.

When extended to *n*-grams, the modified precision is similarly computed; after collecting all the candidate *n*-gram counts and corresponding maximum reference counts, the candidate counts are

again capped by the maximum value, summed, and divided by the total candidate n-grams. This modification to precision p_n means matches made are much likely to be more fluent as they match on a whole sequence of words rather than just a single word unigram approach which more matches the aspect of adequacy.

$$p_n = \frac{\sum_{C \in \text{Candidates}} \sum_{n\text{-gram} \in C} \text{Count}_{clip}(n\text{-gram})}{\sum_{C' \in \text{Candidates}} \sum_{n\text{-gram}' \in C'} \text{Count}(n\text{-gram}')}$$

With this n-gram approach, it was found that the precision of translations decayed exponentially as the value of n increased. To take this into account, the BLEU metric uses a weighted average of the logarithm of modified precisions, which allows combining the different n-gram precisions, from 1-gram to a maximum of 4-gram sizes. This value of a 4-gram maximum was found experimentally, providing the best correlation with monolingual human judgments.

While the modified precision measures already penalise irrelevant words in the candidate that do not appear in the reference as well as overuse of a certain word, it does not penalise translations of the wrong length. The following example illustrates this:

Candidate: A man

Reference: Something must not be working in the house for a man to have this reaction.

Even though the candidate translation has practically no relevance to the reference sentence, it still carries maximal precision, because the phrase “*a man*” appears exactly as written in the reference. The BLEU metric introduces a brevity penalty to deal with this omission of information, a multiplicative factor to reduce the score of the candidate in these situations. The brevity penalty has a value of 1 when the candidate translation’s length is the same as the length of any of the reference translations. As translations with greater length than the reference are already punished for spurious words in the precision calculation, both extremes of a sentence with incorrect length are handled in this measurement.

The brevity penalty BP is calculated over the entire corpus rather than sentence by sentence, to avoid bias against shorter sentences. The penalty is a decaying exponential in r/c , where r is the length of the reference translation and c is the length of the candidate translation.

$$BP = \begin{cases} 1 & c > r \\ e^{1-r/c} & c \leq r \end{cases}$$

Given this, the BLEU score is then calculated taking p_n as the geometric average of the modified n -gram precisions, with n -grams up to length N and positive weights w_n summing to 1. In the standard BLEU metric, $N = 4$ and $w_n = 1/N$.

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

The BLEU metric in general correlates strongly with human judgment, but is not without several weaknesses. The 4-gram standard, for example, is not one based on any objective finding but more a result of experimental perception. That 3-gram and 5-gram maximums have been found to give similar results, as the exponential decay means that the higher orders already have very small values [48]. Restricting the search to 4-gram blocks may mean rewarding a translation that is fluent in chunks of 4 blocks, rather than a holistic approach of assessing the entire text at the same time. The metric also treats each of the blocks with the same uniform weight, rather than weighing more heavily n-grams which provide greater information to a translation. It has also been noted that the correlation for

professional translators is much smaller than for machines [19], even though their scores are distinctly better. This suggests that the differences between professional translators are far more subtle than those that can be analysed through n-gram statistics.

In addition, BLEU solely focuses on precision, and does not take recall into account. Precision identifies the proportion of matched n-grams out of the total number of n-grams in the candidate translation, but recall is centred on the proportion of matched n-grams out of the total number of n-grams in the reference translation. Recall is important in assessing what degree the translation covers the content of the reference sentence, but BLEU uses the brevity penalty to compensate, an approach which can be inadequate [1].

Nevertheless, BLEU remains one of the most popular metrics in this field, with its correlation to human judgments a key factor. Its method of averaging out individual sentence errors over an entire text rather than focusing on the exact wording for each sentence means “quantity leads to quality.” [48]

2.2.3 NIST

An approach to rectifying the issue of all n-grams having equal weight without regard to their quality of information is the NIST metric, which adapts the BLEU metric to introduce a measure of informativeness [19]. The less frequent an n-gram is in the text, the more important its message is, as its uniqueness implies the words carry a distinct meaning. On the other hand, a more frequent n-gram may be comprised of a set of function words that add little to the actual semantics of the text. Take the following example:

Sentence: “Interesting work in the valley means there is better food in the village”

Bigram 1: “in the”

Bigram 2: “interesting work”

The phrase “*in the*” occurs twice, but adds little meaning to the interest of the text, whereas the bigram “*interesting work*” only occurs once and carries a greater importance to the overall meaning. These more frequent n-grams will receive lower information weight in calculating the NIST score.

$$\text{Info}(w_1 \dots w_n) = \log_2 \frac{\# \text{occurrences } w_1 \dots w_{n-1}}{\# \text{occurrences } w_1 \dots w_n}$$

The NIST calculation also adapts the brevity penalty, lower than that of BLEU for candidate translations that are on the shorter side, but reverting to a similar level for lengthier translations. This value of β in the calculation is chosen to make the penalty factor 0.5 when the number of words in the system output is two-thirds of the number of words in the reference. The metric also uses $N = 5$ as standard, with L_{ref} being the average number of words in the reference translation over all the reference translations, and L_{sys} being the number of words in the candidate translation.

$$NIST = \sum_{n=1}^N \left\{ \frac{\sum_{w_1 \dots w_n} \text{Info}(w_1 \dots w_n)}{\sum_{w_1 \dots w_n} 1} \right\} \times \exp \left\{ \beta \log^2 \left\{ \min \frac{L_{sys}}{L_{ref}}, 1 \right\} \right\}$$

The NIST study also looked to analyse the effect of several additional parameters on the metric’s score. While some factors such as the source of the text and the number of reference translations did not have a major impact on correlation, it was found that segment size would impact the performance of the NIST algorithm. Increasing the size of the segment that co-occurrences can be restricted produces poorer performance, as the average number of words in a document increases. However, despite the fact that smaller segments naturally provide better performance, this constraint is difficult to maintain and rather unnatural.

2.2.4 ROUGE

Another metric that seeks to improve on the deficiencies of BLEU is the ROUGE metric [38]. This metric looked to target the subjective brevity penalty, as well as focusing on sentence level structure and dynamic. Based on the idea of the longest common subsequence (LCS), sentence level structure similarity is naturally incorporated, helping to identify co-occurring n-grams. An extension of the algorithm also uses skip-bigrams, a relaxed version of strict n-gram matching.

A sequence $Z = \langle z_1, z_2, \dots, z_n \rangle$ is a subsequence of another sequence $X = \langle x_1, x_2, \dots, x_n \rangle$ if there exists a strict increasing sequence $\langle i_1, i_2, \dots, i_k \rangle$ of indices of X such that for all $j = 1, 2, \dots, k$, $x_{i_j} = z_j$. For example, the sequence $Z = \langle B, C, A, B \rangle$ is a subsequence of $X = \langle A, B, C, D, A, D, B \rangle$ with a corresponding index sequence $\langle 2, 3, 5, 7 \rangle$. Given two sequences X and Y , the sequence Z is a common subsequence of X and Y if it is a sequence of both X and Y . The longest common subsequence is therefore the common subsequence with maximum length.

The ROUGE-L algorithm applies LCS to translation evaluation by treating a translation as a sequence of words. The longer the LCS of two sequences is, the more similar the two sequences are. Applying this to a reference translation r length m and a candidate translation c length n , the metric uses an LCS-based F-measure to estimate the similarity between the two. The unigram-based F-measure has been found to have strong correlation with human judgments [60]. A measure of accuracy, the F-measure uses both precision P and recall R in its calculation, with the parameter β defined as the factor in which recall is weighted above precision as follows:

$$R = \frac{LCS(r, c)}{m}$$

$$P = \frac{LCS(r, c)}{n}$$

$$ROUGE-L = \frac{(1 + \beta^2) \times R \times P}{R + \beta^2 \times P}$$

And if multiple reference translations r_j in a set u , each of m_j words are used with a candidate translation c of n words:

$$R = \max_{j=1}^u \left(\frac{LCS(r_j, c)}{m_j} \right)$$

$$P = \max_{j=1}^u \left(\frac{LCS(r_j, c)}{n} \right)$$

$$ROUGE-L = \frac{(1 + \beta^2) \times R \times P}{R + \beta^2 \times P}$$

A distinct advantage of LCS matching rather than strict n-gram matching is that it does not require consecutive matches of words, but just needs them to be in sequence. This flexibility still preserves the natural sentence level word order as n-grams. It also means there does not need to be a predefined n-gram length, as the method will just automatically take the common n-grams with the longest length. As this method only rewards n-gram matches that are in sequence, it can better differentiate sentences with similar words but vastly opposite meanings, as opposed to the pure n-gram matching of BLEU.

Reference: Police killed the gunman
Candidate 1: Police kill the gunman
Candidate 2: The gunman kill police

In this example, BLEU would not be able to differentiate between the two candidates as it would match on the unigram “*police*” and the bigram “*the gunman*” for both sentences. On the other hand, the ROUGE-L metric would be able to spot the length 3 subsequence “*police the gunman*” for Candidate 1, as opposed to the length 2 subsequence “*the gunman*” for Candidate 2, giving Candidate 1 a better score.

An extra adaptation is the ROUGE-W metric, which better rewards LCS matches that are of consecutive words. Given the following sequences:

X : [A B C D E F G]

Y_1 : [A B C D H I K]

Y_2 : [A H B K C I D]

The candidate Y_1 is clearly a closer fit to the reference X , but has the same ROUGE-L score as Y_2 . A weighted LCS function f used instead of the standard LCS to give greater scores to consecutive matches. Dynamic programming can be used to keep track of what the longest consecutive match k is, which can be used as a parameter in the weighted function, such as $f(k) = k^2$.

$$R = f^{-1} \left(\frac{WLCS(r, c)}{f(m)} \right)$$

$$P = f^{-1} \left(\frac{WLCS(r, c)}{f(n)} \right)$$

$$ROUGE-W = \frac{(1 + \beta^2) \times R \times P}{R + \beta^2 \times P}$$

The ROUGE-S extension to this metric uses skip-bigrams as a match rather than LCS. This better targets shorter common sequences that are ignored by the LCS approach, but still retains the flexibility of having non-consecutive matches. A skip-bigram is any pair of words in the sentence order, allowing for arbitrary gaps. For example, in the sentence “*Police killed the gunman*”, there are 6 possible skip-bigrams: “*police killed*”, “*police the*”, “*police gunman*”, “*killed the*”, “*killed gunman*”, “*the gunman*”. The skip-bigram-based measure compares overlap of these bigrams between the reference and candidate translations, where S is the number of skip-bigram matches, m is the length of the reference r and n is the length of the candidate c :

$$R = \frac{SKIP2(r, c)}{C(m, 2)}$$

$$P = \frac{SKIP2(r, c)}{C(n, 2)}$$

$$ROUGE-S = \frac{(1 + \beta^2) \times R \times P}{R + \beta^2 \times P}$$

The ROUGE metric offers results that bring an improvement on the BLEU and NIST metrics both in fluency and adequacy. Focusing on non-consecutive matching, the metric allows for more relevant information to be utilised, as the criteria is not as strict. This means sentence level word orders and structures are more respected, making the metric far more applicable to the sentence level rather than the corpus-focused style of BLEU. The focus on recall also eliminates the rather arbitrary brevity penalty. However, the metric is still only applying string matching, and does not focus on the intrinsic meanings of words.

2.2.5 METEOR

METEOR [1] is a metric based on flexible unigram matching. It differs from other metrics by including matching based on word stems and synonyms, allowing words that are simple variants of a stem word to be matched where other algorithms would ignore it. Utilising unigram precision, unigram recall and a measure of fragmentation in a sentence’s word order, METEOR calculates a score that can produce a good correlation with human reference at both sentence and corpus level via explicit matching word-to-word of a reference and candidate translation.

The METEOR metric was created to target several flaws in the BLEU baseline; the lack of recall and explicit word matching, the use of geometric averages of n-grams, and using higher-order n-grams to indirectly assess fluency instead of an explicit measure of word order and “grammaticality”. The metric goes through two phases, the first to align unigrams to their counterparts, the second to select the best alignment. These two phases are repeated for each stage that the algorithm uses; each stage representing the module used to select the alignments. These stages are run consecutively, starting from the most straightforward exact word matching, progressing to word stem matching, and then to synonym matching. Each iteration only appends extra mappings to the alignment, so the order in which stages are run is a reflection of the priority of each.

The first phase of the METEOR algorithm is alignment. An alignment is a mapping between unigrams, so that every unigram in each string maps to zero or one unigram in the opposite string, and none in the same string. In any alignment, a single unigram in one string cannot map to more than one unigram in the opposite string, but can be mapped to by multiple unigrams. The selection of mappings depends on the module used. The exact match module maps words onto another only if they are exactly the same. For example, “*blouse*” would map to “*blouse*” but would not map to “*blouses*”. The stemmer module maps words onto another if they come from the same stem word, so “*friendship*” would map to “*friendship*” as well as “*friendships*” as they both come from the same stem. A stem is the part of the word which is common to all the inflected variants [31], so this example would not match on the word “*friend*”, which is instead the root of the word. The synonym module matches words if they are synonyms of each other; so “*good*” would map to “*well*”, but would not have any link to “*mediocre*”.



Figure 2.2: Two alignments of the same reference and candidate sentence, with the same number of mappings. The alignment on the left would be selected as it has fewer unigram mapping crossings.

The second phase of METEOR involves selecting the right alignment, which is generally the one with the largest amount of mappings. If there are two alignments with the same number of mappings, the set with the fewest unigram mapping crosses is selected. If the two strings are written one on top of the other and mappings are signified with a line between two words, each line crossing is a “unigram mapping cross”. Formally, two unigram mappings (c_i, r_j) and (c_k, r_l) , where c_i and c_k are unigrams in the candidate translation and r_j and r_l are unigrams in the reference translation, are said to cross if the following evaluates to a negative number, with $pos(x)$ the index of the unigram x in the string.:

$$(pos(c_i) - pos(c_k)) \times (pos(r_j) - pos(r_l))$$

The METEOR score is calculated using the precision and recall of the selected mapping, as well as a fragmentation penalty to account for the translation’s fluency.

Unigram precision P is calculated as:

$$P = \frac{m}{w_c}$$

Unigram recall R is calculated as:

$$R = \frac{m}{w_r}$$

where m is the number of unigrams in the candidate translation also found in the reference translation, w_c is the number of unigrams in the candidate translation, and w_r is the number of unigrams in the reference translation. These two values are combined with the harmonic mean, where recall is weighted 9 times more than precision [52]:

$$F_{mean} = \frac{10 \times P \times R}{R + 9P}$$

In addition, a fragmentation penalty is introduced to assess the congruity of the candidate translation. The more mappings that are not adjacent in the candidate and reference sentence, the higher the penalty is. To compute this, the unigrams are grouped into chunks, where a chunk is defined as a group of unigrams that are adjacent in both the candidate and reference translation. The longer the n-grams, the fewer the chunks, so if the entire candidate translation matches the reference there is only one chunk. The penalty is computed as:

$$\text{Penalty} = 0.5 \times \left(\frac{c}{u_m} \right)^3$$

where c is the number of chunks and u_m is the number of unigrams that have been matched. The penalty increases with the number of chunks to a maximum of 0.5.

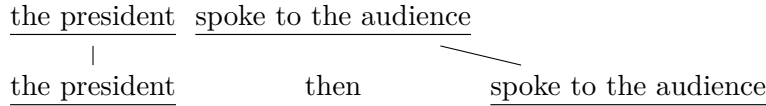


Figure 2.3: An example of two chunks in two sentences.

The METEOR score for a given alignment is finally given as:

$$M = F_{mean} \times (1 - \text{Penalty})$$

METEOR has also been improved in further extensions. The metric has been extended to different languages [33], tuning different parameter values to different purposes [34]. A length penalty has also been introduced to prevent high recall and low precision sentences from receiving a high score [35], as well as text normalisation [16] and changes the alignment to include paraphrase matching and optimisation for the post-editing HTER metric [17].

METEOR has been shown to outperform BLEU at correlation with human judgments, making it a useful tool in automatic machine translation evaluation. However, it still contains some of the inherent weaknesses of a pattern matching system, which have to be finely tuned via several parameters to balance it. The weighting of precision over recall as well as the restriction of the penalty to 50% are all figures which appear arbitrary, but have gone through many levels of experimentation to reach. The matching of unigrams is also done on a very static basis; words are only matched if they have been explicitly labelled as a synonym or a paraphrase in the relevant table. This does not fully embrace the semantic relatedness of each word, which limits the effectiveness of the metric as two sentences may not be similar in word form, but can carry the same meaning.

2.3 Distributional Semantics

Distributional semantics are a novel approach of quantifying words and their semantic meanings. This is based on the distributional hypothesis, which suggests that linguistic items with similar distributions have similar meanings [28], or “a word is characterised by the company it keeps” [22]. In principle, words that appear next to “*dog*” are much more likely to also appear next to “*cat*”, as opposed to “*coconut*”. There are two main styles to achieve these language models: count-based methods and predictive methods. Both of these styles leverage the distributional hypothesis to create word embeddings, mapping words and phrases to vectors in a semantic space, with each dimension in the space representing a different feature of the word. These embeddings can be used in translation evaluation, taking semantic relatedness into account rather than syntactic string matching.

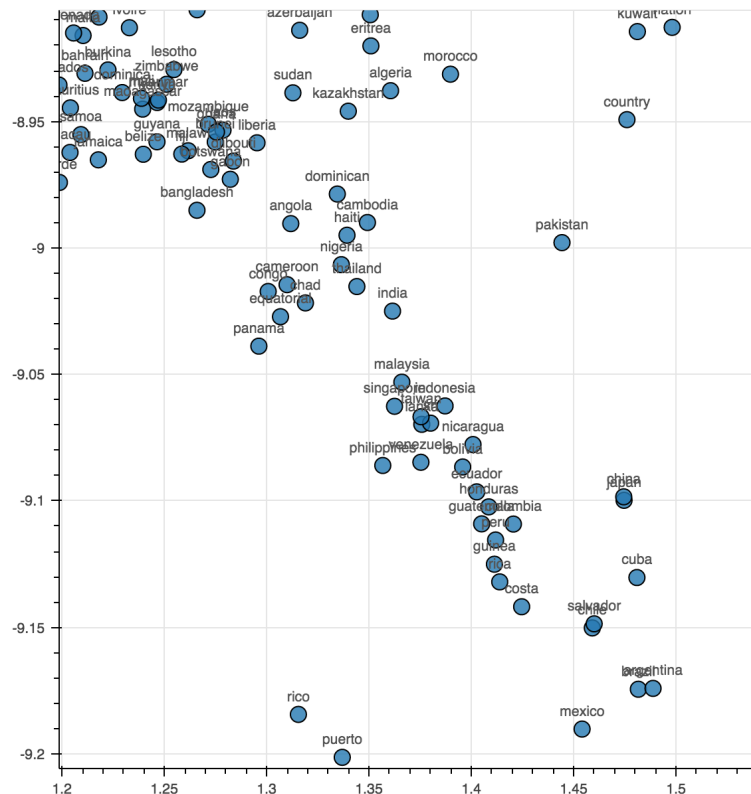


Figure 2.4: An example of different country name word vectors, using t-SNE representation to display the high-dimensional data in 2D. Note how similar countries cluster together; the top left corner predominantly African countries, the bottom right primarily Latin American countries [27].

2.3.1 Count-based methods

In count-based methods, words are expressed as vectors of co-occurring words. The methods compute statistics of how often a word appears in the same context as its neighbouring words, and maps these to a small and dense vector for each word. To do this, a large number of training corpuses are used to collect information about occurrence of words. The technique takes a space where each word is represented by one dimension and embeds this into a continuous vector space with a much smaller dimension. One can imagine that each dimension corresponds to a semantic or grammatical characteristic. This allows words of a similar context to be in close spatial proximity, allowing the use of various distance metrics to measure semantic similarity when evaluating translations, so words can be meaningfully compared even if they never appear together in a document.

Latent semantic analysis (LSA) is a technique that represents these word counts as a sparse matrix, using singular value decomposition to reduce the number of rows while maintaining the similarity structure in its columns. Using a term-document matrix, LSA represents terms in rows and documents in columns. The values assigned to each cell are generally an indicator of their occurrence in the texts, but it has been found that weightings towards informativeness and importance work far better than raw co-occurrence counts [2]. A typical example of weighting is term frequency-inverse document frequency (tf-idf), where weight is proportional to the number of times the word occurs in each document and offset by the number of documents it occurs in. This approach allows rare terms to be upweighted to reflect their importance, as well as adjust for the fact that some words appear more frequently in general.

LSA then uses singular value decomposition (SVD) to create a semantic space with much lower dimension, allowing words to have projections on each other in shared dimensions, rather than having a dimension for each unique term [15]. The procedure of SVD is a simple factorisation of any $m \times n$ matrix M , resulting in the form:

$$M = U\Sigma V^*$$

where U is an $m \times m$ unitary matrix, Σ is a diagonal $m \times n$ matrix with non-negative real numbers (singular values) on the diagonal, V is an $n \times n$ unitary matrix, and V^* is its conjugate transpose.

Applying this to the $m \times n$ term-document matrix X , where element (i, j) is the occurrence of term i in document j , the following formula is reached:

$$\begin{array}{c} X \\ \left[\begin{array}{ccccc} x_{1,1} & \dots & x_{1,j} & \dots & x_{1,n} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{i,1} & \dots & x_{i,j} & \dots & x_{i,n} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{m,1} & \dots & x_{m,j} & \dots & x_{m,n} \end{array} \right] \end{array} = \begin{array}{c} U \\ \left[\left[\begin{array}{c} \mathbf{u}_1 \\ \vdots \\ \mathbf{u}_l \end{array} \right] \dots \left[\begin{array}{c} \mathbf{u}_1 \\ \vdots \\ \mathbf{u}_l \end{array} \right] \right] \end{array} \cdot \begin{array}{c} \Sigma \\ \left[\begin{array}{ccc} \sigma_1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \sigma_l \end{array} \right] \end{array} \cdot \begin{array}{c} V^* \\ \left[\left[\begin{array}{c} \mathbf{v}_1 \\ \vdots \\ \mathbf{v}_l \end{array} \right] \right] \end{array}$$

Each singular value in the diagonal matrix Σ has corresponding vectors in U and V , which are used in compressing the matrix into a smaller representation. By convention, the diagonal values are ordered by descending value, giving an indication of the amount of impact each has on the behaviour of the matrix. If decomposed matrix is reduced to just the top k singular values and their corresponding vectors, a least squared error approximation to the original matrix is achieved using a smaller set of numbers [24], discarding the values which did not have a large impact on the overall embedding.

This rank k approximation to X can be represented as such:

$$X_k = U_k \Sigma_k V_k^*$$

Documents j and q can then be compared in this low-dimensional space by comparing vectors $\Sigma_k d_j$ and $\Sigma_k d_q$, where d_j and d_q are columns in V_k^* . Terms i and p can be similarly compared by comparing vectors $\Sigma_k t_i$ and $\Sigma_k t_p$, where t_i and t_p are transposes of the rows in U . Given a query q , it can also be compared to a document by translating the query to the low-dimensional space and using the same comparison one would with documents, or vice versa with terms as follows:

$$q = \Sigma_k^{-1} U_k^T \mathbf{d}_j$$

$$q = \Sigma_k^{-1} V_k^T \mathbf{t}_i$$

Count-based methods such as LSA are well-established means to create word embeddings, providing a simple and automatic way to construct these semantic spaces. However, these methods have been found to be outperformed by newer predictive models [2].

2.3.2 Predictive methods

Predictive models are language models that try to directly predict a word from its neighbours, using learned word embeddings. These types of language models have been brought about by the growth of machine learning techniques, which put these vector estimation problems in the context of a supervised task. Since similar words occur in similar contexts, the system naturally learns to assign similar vectors to similar words. These feature vectors can also be used in translation evaluation, plotting each word's meaning in a semantic space.

The traditional approach to probabilistic language models is based on n-grams [29], storing and combining frequency counts for word subsequences of different lengths to estimate the likelihood of each possible next word. In an n-gram model, the probability of observing a sentence (w_1, \dots, w_m) is approximated as [13]:

$$\begin{aligned} P(w_1, \dots, w_m) &= \prod_{i=1}^m P(w_i | w_1, \dots, w_{i-1}) \\ &\approx \prod_{i=1}^m P(w_i | w_{i-(n-1)}, \dots, w_{i-1}) \end{aligned}$$

The assumption is made that the probability of observing the i^{th} word w_i given the context of the previous w_{i-1} words can be approximated by just taking the shortened context history of the previous $n - 1$ words instead. This conditional probability can be calculated simply from frequency counts, but these are often altered with various smoothing techniques to account for unseen n-grams.

$$P(w_i | w_{i-(n-1)}, \dots, w_{i-1}) = \frac{\text{count}(w_{i-(n-1)}, \dots, w_{i-1}, w_i)}{\text{count}(w_{i-(n-1)}, \dots, w_{i-1})}$$

Modern predictive models use neural networks to learn distributed representations, utilising their ability to interchange functionally similar words to compactly represent a function that makes good predictions [3]. The advantage of this approach is that the model can generalise well to sequences not in the training set that still have similar features. As neural networks tend to map nearby inputs to nearby outputs, the predictions that correspond to word sequences with similar features are mapped to similar predictions. The compact representation of all the different combinations of feature values also limits the curse of dimensionality, allowing the model to fit a large training set. Neural networks also allow more complexity in the model, implementing characteristics like similarity, a contrast to the simple atomic approach of n-grams.

Neural networks are a machine learning architecture that essentially transform numerical vectorised input x to a numerical vectorised output y via a parameterised function f . The basic unit of these networks is the neuron, which receives inputs and produces outputs. These inputs are weighted by learned parameters, and are generally linked to one another in layers. A feedforward neural network will successively apply these transformations, whereas a recurrent neural network may have layers that apply transformations to themselves. These layers include the input and output layer, as well as multiple hidden layers. There can also exist a projection layer, which maps discrete indices from the input to a continuous vector space for the hidden layers.

In contrast with the traditional n-gram predictive models, a distributed representation contains many neurons active at the same time to classify an object efficiently. For example, if m binary features are used then 2^m distinct objects can be described. On the other hand, an n-gram model uses a local representation, where only units associated with the specific subsequences of the input sequence are turned on, which grows exponentially with sequence length, causing data sparsity.

Neural network models also find the probability of a sentence by calculating the conditional probability of each word given its predecessors [4], where a feedforward neural network with a linear projection layer and a non-linear hidden layer is used to jointly learn the word vectors and a statistical language

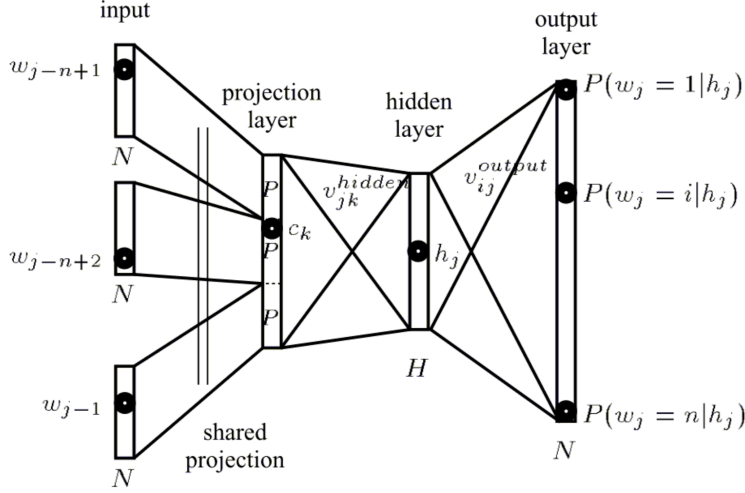


Figure 2.5: An example feedforward neural network architecture [46]

model. The probability $P(w_t|w_{t-(n-1)}, \dots, w_{t-1})$ in the context of sentence (w_1, \dots, w_t) is obtained using a parameter matrix C containing learned features of words. Each word w_{t-i} in the $n-1$ word context preceding w_t is mapped to a d -dimensional feature vector $C_{w_{t-i}}$, which is column w_{t-i} of C , containing the learned features for that word. The $n-1$ feature vectors are then concatenated into the vector x :

$$x = (C_{w_{t-(n-1)},1}, \dots, C_{w_{t-(n-1)},d}, C_{w_{t-(n-2)},1}, \dots, C_{w_{t-2},d}, C_{w_{t-1},1}, \dots, C_{w_{t-1},d})$$

The prediction of the next word being k is then obtained as follows, using the softmax activation function on the output neurons [5]:

$$P(w_t = k|w_{t-(n-1)}, \dots, w_{t-1}) = \frac{e^{a_k}}{\sum_{l=1}^N e^{a_l}}$$

where

$$a_k = b_k + \sum_{i=1}^h W_{ki} \tanh(c_i + \sum_{j=1}^{(n-1)d} V_{ij} x_j)$$

and vectors b, c along with matrices W, V are also parameters. The capacity of this model is controlled by the number of hidden neurons h and the number of learned features d . Given θ as the concatenation of all parameters, the neural network is trained to maximise the training set log-likelihood:

$$L(\theta) = \sum_t \log P(w_t|w_{t-(n+1)}, \dots, w_{t-1})$$

Because of the large number of examples, training the model is not a straightforward task. This is both slow and expensive, as there is a need to compute and normalise each probability using the score for all the other words in the current context at every single training step. Optimised algorithms are often used to avert this, including only using the neural network for a subset of words [56] or caching softmax normalisation constants [67]. Some approaches also transform the model to be hierarchical, using binary trees to increase efficiency and reduce training time [47]. Nevertheless, these predictive models are able to replace the heuristic vector transforms of count-based models with a single well-defined supervised learning step, while using the same data, and do so with far more encouraging results.

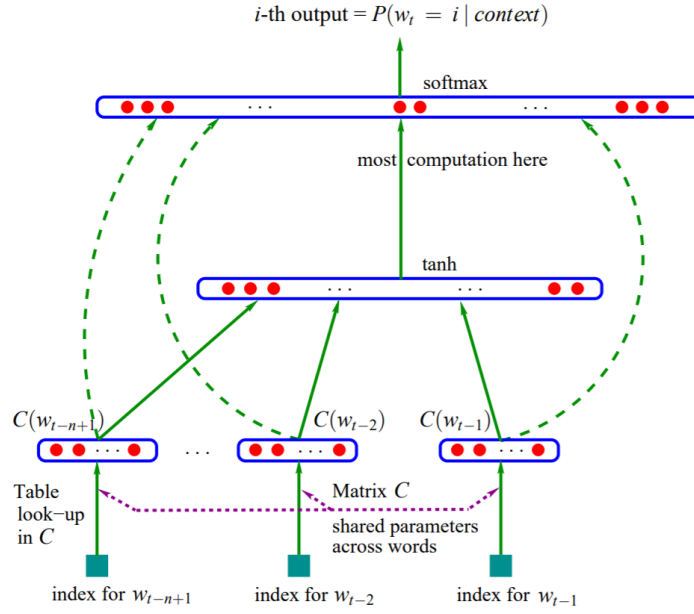


Figure 2.6: The neural architecture $f(i, w_{t-1}, \dots, w_{t-(n-1)}) = g(i, C(w_{t-1}), \dots, C(w_{t-(n-1)}))$ where g is the neural network and $C(i)$ is the i -th word feature vector, specified by [4].

2.3.3 Word2vec

At the forefront of the current state-of-the-art in distributional semantics, word2vec is a model created by [45] that produces high quality word embeddings at much lower computational cost. The word2vec model introduces new techniques to train vectors over billions of words, a vast improvement on the hundreds of millions that other neural techniques can operate on. Word2vec adapts the work of previous neural network language models, focusing on the interim step where word vectors are learned without actually constructing the full language model [42].

Word2vec can utilise either of two architectures to produce word embeddings: continuous-bag-of-words (CBOW) or continuous skip-gram. The CBOW architecture predicts the current word given the context, whereas skip-gram predicts the surrounding words given the current word. The bag-of-words approach ignores the order of words in the context, but the skip-gram approach will give more weight to words closer to the current word by sampling less from words that are distant.

The CBOW model is similar to the feedforward neural network language model. The non-linear hidden layer is removed and the projection layer is shared for all words, so all words get projected to the same position and order is irrelevant. In contrast to other models, words from the future are also used along with previous words to predict the current word. The skip-gram model uses each current word as the input to the classifier, which has a continuous projection layer. The complexity of these architectures are both lower than other neural network language models, as most of the complexity from those models comes from the non-linear hidden layer, which is removed in these simple models. While data may not be able to be represented as precisely, training is made far more efficient. The training complexity Q of CBOW is:

$$Q = N \times D + D \times \log_2(V)$$

while the training complexity of skip-gram is:

$$Q = C \times (D + D \times \log_2(V))$$

where $N \times D$ is the dimensionality of the projection layer, V is the size of the vocabulary and C is the maximum distance of words to use in the skip-gram approach.

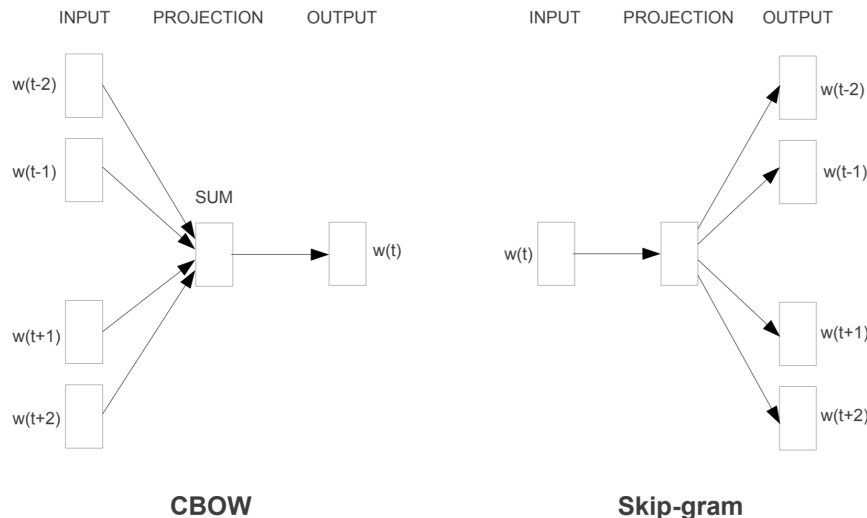


Figure 2.7: The two model architectures of [45]. The CBOW architecture predicts the current word based on the context, and the skip-gram predicts surrounding words given the current word.

The word2vec architecture provides a high quality vector representation for words while allowing far more efficient training. High dimensionality vectors are able to be trained on a large amount of data, several orders of magnitude greater than other existing models as a result of the simple architecture. The accuracy of these vectors are able to detect very subtle semantic relationships between words, creating a nuanced understanding of word features. In a well-trained set of vectors, relationships such as country and city can be targeted using simple vector algebra. For example, to find what city is to Germany as Paris is to France, the calculation $X = \text{vector}(\text{"France"}) - \text{vector}(\text{"Paris"}) + \text{vector}(\text{"Germany"})$ returns the word *"Berlin"* as X .

The model has some weaknesses in its composition. Words are matched exactly rather than accounting for synonyms and stemmed phrases, meaning word morphology is ignored. Since every word is treated as an independent vector, there are no common representations at sub-word levels. This can make learning embeddings more difficult in polysynthetic languages, such as Arabic or German. Scaling to a new language also means new embedding matrices, such that cross-language sharing of the same parameters in the model is not feasible. Furthermore, word2vec is unable to handle unknown words that are not part of the training vocabulary. If asked to define a vector for a word that has never been encountered before, the model cannot provide a meaningful answer; it would have to discard the request or assign a random value. As with other word embeddings, polysemy is neglected as all possible meanings for a single word are combined into one vector. This can be targeted using sense embeddings, where each meaning of a word is its own vector in the semantic space.

The training performance of word2vec is also heavily dependent on hyper-parameters selected [65]. In terms of the two architectures, CBOW provides a faster training time, but skip-gram works better for infrequent words. The training algorithm to maximise the log-likelihood can also be selected between the hierarchical softmax, which is better for these infrequent words, and negative sampling, which works better for frequent words and low dimensional vectors. Frequent words can also be subsampled to increase training speed [44], while dimensionality can also be increased to provide higher quality vectors. Up to a certain point, however, the quality gain experienced by increasing the dimension will diminish. The size of the context window also has an affect on performance; according to the authors the recommended size is 10 for skip-gram and 5 for CBOW. It has also been suggested much of the accuracy of word2vec and similar embeddings are not necessarily because of the models, but a result of the choice of hyper-parameters [37].

2.3.4 fastText

fastText [Bojanowski:2017] is a different library for efficient text classification and learning word representations. An open-source and lightweight distribution, fastText builds upon standard word vectors by including subword information to create a word representation for any given token, even those not part of the embedding’s vocabulary. This targets one of the weaknesses of the word2vec method, which uses a distinct vector for each word and ignores the internal structure of a word.

fastText supports supervised and unsupervised training, as well as both skip-gram and CBOW models. It is able to achieve strong performance for word representations, particularly for rare word occurrences that utilise character n-grams. Words in the vocabulary have their own embeddings, as well as embeddings of the n-grams which make up the word. The length of n-grams is an adjustable parameter that affects the final embedding result. Given the word “table” and an n-gram size 3, where “<” and “>” are characters to mark the start and end of a word, the associated bag of n-gram characters will consist of:

“<ta”, “tab”, “abl”, “ble”, “le>”

Note that the n-gram representation of “tab” here will be different from the vector representation of the word “tab”, as one is a component of a word and one is a word representation itself.

The vector representation of a single word is then simply the sum of all the vector representations in the bag of character n-grams. This approach allows innate meanings in prefixes and suffixes to be captured and re-used for different words. It also means that words with similar forms will naturally have similar embeddings; for example “worker” will have a very similar vector representation as “workers”, as nearly all of the n-grams will be identical – even if one of the words is not part of the actual vocabulary of the word vectors.

In terms of training the fastText model, several optimisations are in place to better performance, which can also double up as limitations in terms of vector quality. For example, the training algorithm prunes the vocabulary every time the vocabulary exceeds a hard coded size, by increasing the provided minimum count value. Negative sampling can also be used to speed up training, as can discarding of popular words to prevent saturation of embeddings. Training can also be initialised with pre-trained vectors, allowing a trained embedding to build on top of an existing set of vectors [Subedi:2018].

This method proves to be fast in calculation and is able to train on large corpora quickly, while having strong performance in word similarity and analogy tasks comparable to word2vec and other similar embedding styles.

2.4 Distance metrics

Given the vector space models provided by word embeddings, various distance metrics for vectors can be used to measure the similarity of words, and thereby create a similarity metric to assess how translations match up against each other. This can be applied to translation evaluation, comparing how candidate translations match up against reference translations.

2.4.1 Cosine similarity

Cosine similarity is a measure of similarity between two non-zero vectors, measuring the cosine of the angle between them in the vector space. This is a measure of orientation rather than magnitude; two vectors with the same orientation have a cosine similarity of 1, orthogonal vectors have a similarity of 0, while diametrically opposed vectors will return -1. This is all independent of magnitude, and applicable to any number of dimensions. With word embeddings, each dimension represents a different feature, so if two vectors are oriented similarly, it means they are aligned on many dimensions and

therefore have similar semantic features. The cosine similarity $\cos(\theta)$ is defined as:

$$\cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|} = \frac{\sum_{i=1}^n A_i B_i}{\sqrt{\sum_{i=1}^n A_i^2} \sqrt{\sum_{i=1}^n B_i^2}}$$

where A_i and B_i are components of vector \mathbf{A} and \mathbf{B} respectively.

The soft cosine measure is a variant of this metric, considering similarities between pairs of features [57]. While the standard cosine similarity considers each feature as independent and completely different, the soft cosine measure relates features by their similarity. A matrix s can be used to indicate similarity between features, making the distinction between each dimension less harsh. If there is no similarity between features, the calculation is the equivalent of the conventional cosine similarity. Given two N -dimensional vectors a and b and the similarity matrix s , the soft cosine similarity is calculated as:

$$\text{soft_cosine}(a, b) = \frac{\sum_{i,j} s_{ij} a_i b_j}{\sqrt{\sum_{i,j} s_{ij} a_i a_j} \sqrt{\sum_{i,j} s_{ij} b_i b_j}}$$

where $s_{ij} = \text{similarity}(\text{feature}_i, \text{feature}_j)$.

An advantage of this metric is its low complexity, especially for sparse vectors as only non-zero dimensions need to be considered when calculating the angle. This works very well with the hundreds of dimensions that a word embedding vector possesses. However, the metric is only useful if the magnitude of the vectors do not matter. Vectors are often normalised to use this metric effectively, but word embeddings from models like word2vec carry semantic significance in a vector's length as well as the direction [55]. Nevertheless, it must be noted that the original word2vec paper [45] still uses cosine similarity as a measure of finding similar words in the semantic space.

2.4.2 Euclidean distance

The Euclidean distance is a measure of straight-line distance between two points in a Euclidean space. This is akin to taking a ruler and measuring the length of the line between two points. In vector spaces, these two points are defined by corresponding vectors \mathbf{A} and \mathbf{B} , with a component A_i and B_i for each dimension. The Euclidean distance can be extended to any number of dimensions n , defined as follows:

$$d(\mathbf{A}, \mathbf{B}) = \sqrt{\sum_{i=1}^N (A_i - B_i)^2}$$

Given vectors are a representation of direction as well as magnitude, the use of Euclidean distance can be an appropriate metric for finding distances within the word embedding space. However, this may not be as optimal for sparse data, as many calculations would have to take place to handle high dimensionality.

2.4.3 Earth Mover's Distance

The Earth Mover's Distance (EMD) is a measure of the distance between two probability distributions over a region D . Envisioning each distribution as a pile of earth in the region D , the EMD is the minimum cost of turning one pile into the other, where the cost is the amount of earth multiplied by the distance moved. If the two distributions are of the same mass, i.e. the two piles are of the same size, the EMD is a measure of true distance between the distributions. This can be formulated as a transportation problem, and solved with a host of established techniques. As a result, EMD has many applications in pattern recognition [54].

To represent a distribution, EMD generally uses the notion of a signature. A signature contains a set of clusters and their corresponding weights. Each cluster comprised of a set of points, represented by a single average point. The cluster can be thought of as an individual feature of the signature, while the weight is the fraction of the distribution that the cluster represents. The distance between each of these clusters is the ground distance. As a result, simple distributions may have shorter signatures than complex ones.

Given two signatures P with m clusters $(p_1, w_{p1}), \dots, (p_m, w_{pm})$ and Q with n clusters $(q_1, w_{q1}), \dots, (q_n, w_{qn})$, where p_i and q_j are the cluster points and w_{pi} and w_{qj} are their corresponding weights, flow is found between the two signatures that minimises total cost. $D = [d_{i,j}]$ is the ground distance between clusters p_i and q_j , while $F = [f_{i,j}]$ represents the flow between them. The problem is modelled as a linear programming optimisation:

$$\min \sum_{i=1}^m \sum_{j=1}^n f_{i,j} d_{i,j}$$

subject to constraints:

$$\begin{aligned} f_{i,j} &\geq 0, 1 \leq i \leq m, 1 \leq j \leq n \\ \sum_{j=1}^n f_{i,j} &\leq w_{pi}, 1 \leq i \leq m \\ \sum_{i=1}^m f_{i,j} &\leq w_{qj}, 1 \leq j \leq n \\ \sum_{i=1}^m \sum_{j=1}^n f_{i,j} &= \min \left\{ \sum_{i=1}^m w_{pi}, \sum_{j=1}^n w_{qj} \right\} \end{aligned}$$

The first constraint permits moving from P to Q and not vice versa. The next two constraints limit sending and receiving in P and Q by their weights, while the last constraint forces moving the maximum amount of supplies possible. With optimal flow F , the EMD is calculated by normalising the minimum cost by the total flow:

$$\text{EMD}(P, Q) = \frac{\sum_{i=1}^m \sum_{j=1}^n f_{i,j} d_{i,j}}{\sum_{i=1}^m \sum_{j=1}^n f_{i,j}}$$

EMD is a metric that naturally extends the idea of distance from that of single elements to that of distributions of elements. The use of signatures make the measure more compact and efficient than using histograms, which can have sparse and empty bins. When ground distance is meaningfully defined, EMD also matches perceptual similarity much better than other measures [53].

While EMD does not naturally handle distributions with different mass, there are approaches to mitigate this. One is to allow partial matches, where the smaller distribution is compared to a subset of the larger distribution, discarding the extra “earth” at no extra cost [53]. This approach means EMD is not a measure of true distance. An alternative is to create or destroy mass in a distribution rather than transporting it, but incurring a cost penalty. To improve speed and robustness, ground distances may also be thresholded, reducing the edges in the flow network by an order of magnitude [50].

2.4.4 Word Mover’s Distance

An implementation of EMD, the Word Mover’s Distance (WMD) measures the similarity between two text documents. Based on the word2vec embedding space, the metric treats each document as a

distribution of word embeddings, and finds the minimum distance that one set of word embeddings needs to travel to become the other.

To transform one document to another, WMD represents text documents as normalised bag-of-words (nBOW) vectors. This vector d is a column vector of dimension n , where n is the size of the word embedding vocabulary. If word i appears c_i times in the document, the i -th component d_i in d is denoted as $\frac{c_i}{\sum_{j=1}^n c_j}$. WMD removes stop words from this calculation, making it a very sparse vector.

The word travel cost between individual word pairs i and j is represented as $c(i, j)$, a measure of the semantic similarity between the two words. Measures such as the Euclidean distance in the word2vec space can be used to obtain the cost of “travelling” from one word to another. The distance of two words is the basis of finding a cost for two documents.

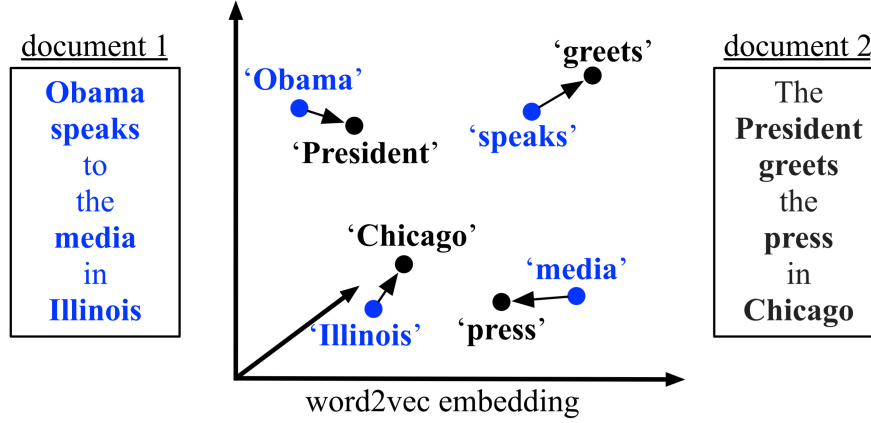


Figure 2.8: An illustration of WMD. The non-stop words of both documents, in bold, are embedded onto a word2vec space. The distance between the two documents is defined as the minimum distance to move all the words from document 1 to exactly match document 2.

Given two nBOW representations d and d' , each word in d' can be transformed into any word in d either by a fraction or in its entirety. The composition of the metric is very similar to that of EMD described above. The cost $c(i, j)$ is a ground distance, while the flow F is here represented as T . The nBOW components d_i and d'_j act as weights of the clusters, where each cluster's position is given by their position in the word2vec embedding space. In the constraints, rather than limiting the amount sent and received by these weights, they are made to send and receive the exact amount specified by the nBOW vector. This is so that all words are matched with the right frequency, instead of over- or under-compensating for words. WMD does not need any extra normalisation after solving the transportation problem, so can be obtained by solution of the linear program:

$$\text{WMD}(d, d') = \min_{T \geq 0} \sum_{i,j=1}^n T_{ij} c(i, j)$$

subject to:

$$\sum_{j=1}^n T_{ij} = d_i, \forall i \in 1, \dots, n$$

$$\sum_{i=1}^n T_{ij} = d'_j, \forall j \in 1, \dots, n$$

The performance of WMD has been shown to have low error rates, attributed to its ability to complement the large scale of the word2vec embedding. This coupling also allows high interpretability

of WMD results, as document distances ultimately boil down to word distances within this semantic space. Not having any hyper-parameters, the metric has very good out-of-the-box functionality.

However, it can be very slow to compute, with a cubic average time complexity [32]. Cheap lower bounds can be used to relax the optimisation problem, helping instances with many unique words or a large number of documents. The nBOW approach of WMD also means word order is not taken into account when finding distances, with no penalty for words being in the wrong position, even if this distorts the overall meaning of a document. Without normalisation, the WMD result is a distance that ranges from 0 to infinity, making comparison between two different calculations of the metric unstandardised.

Part II

Method Overview and Implementation

Chapter 3

Implementation

This project’s automated translation evaluation metric is focused on semantic similarity, giving it a different approach to automated metrics that focus on syntactic matches. In principle, this should allow it to better take into account the overall meaning of a sentence and its translation as the definitions of relevant words will be embedded in the calculation. This is achieved with various neural word distributional models. Appropriate vector space distance metrics are applied to measure word similarity.

The implementation uses the Word Mover’s Distance as its basis. To improve it, modified versions of this metric are pursued to provide a metric that matches the multifaceted human approach of adequacy and fluency, introducing penalties for words being matched in the wrong order or wrong position. This will be referred to as WMD_O , in reference to its focus on word order.

To assess the metric’s performance, direct comparison with human judgment should prove the most conclusive, given human assessment remains the gold standard of translation evaluation. Taking human annotator reviews of existing translations as the benchmark, our metric can be compared to the performance of other metrics in how close they align with the ratings of the human annotators. By creating a metric that is consistently aligned with human judgment of translation quality, it creates a strong basis for the end goal, direct comparison of translation quality between a source text in one language and a translated text in a second language. If reliable, the metric seeks to replace the position of human annotators as the standard, making the evaluation process far more automated. Each rating compares the output of the candidate translation with that of the reference, giving a value of how true to the reference it is. All of these comparisons are in the same language (language B in the diagram).

This section looks at rudimentary experiments using French and German translations, followed by a more comprehensive overview using English as the main language. These are the languages that the source text is translated into, “Language B” in Figure 5.1.

Given a reference translation and a candidate translation in Language B, the flow of the metric implementation is as follows:

3.1 Reference translations

The reference and candidate translations that are used in the implementation carry corresponding human ratings for quality of translation. These ratings are numeric and standardised to z-values for each dataset, making it dimensionless and simple to compare with other datasets. The use of ratings over rankings means translations of similar quality can be given a similar result, rather than picking one over the other.

To statistically compare each metric’s performance against the human score, Pearson’s correlation coefficient is used. This measures the extent in which two variables tend to change together, providing a value for the linear strength and direction of the relationship. Ranging between -1 and 1, a positive

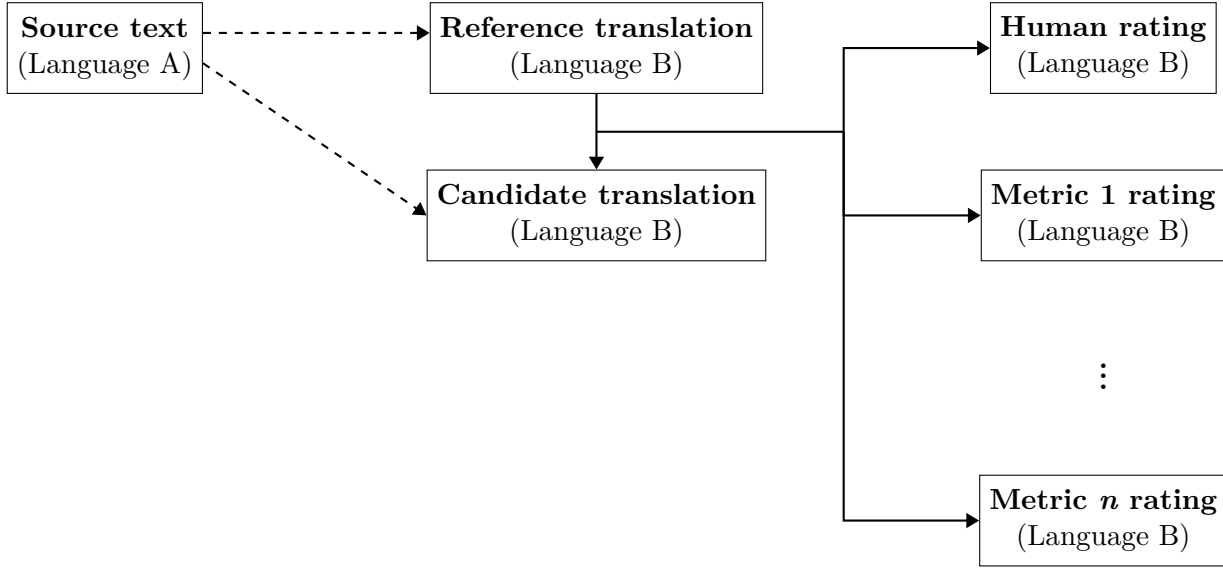


Figure 3.1: Flow of metric evaluations.

value indicates a positive relationship and a negative value a negative relationship. The strength of this measure is that the ranges of the two variables do not matter as long as they vary consistently, meaning the scale of the metrics are not important.

While Pearson’s correlation measures the linear relationship between two variables, the Spearman’s correlation coefficient is an alternative that measures the monotonic relationship between variables. This statistic looks at the correlation between the ranks of each value. However, the relationship between the human scores and the evaluation metric scores are expected to be linear as scores will generally increase linearly with quality. As a result, the Spearman correlation is likely not to be too deviant from the Pearson correlation and will not be pursued heavily.

3.1.1 Datasets

All datasets used for the implementation come from the WMT 2017 conference. The French and German experiments use data from the Multimodal translation task [8], while the English experimental data comes from the Metrics evaluation task [7]. The source language of the former is English, while the latter has different source languages of Czech, German, Finnish, Latvian, Russian, Turkish, and Chinese. These texts are focused on the segment-level rather than the system-level, making each sentence factor more in evaluation. The different source languages are separated when trialling the metric implementation as the human scores are standardised per dataset.

French and German Data

Data from the Multimodal translation task aims to take a source language image description and translate it into a target language, using the image itself to make it multimodal. As the task was not intended for comparing metrics, there is no data for the performance of other metrics on these translations. Each image has a gold standard caption and a set of candidate translations rated by humans from a scale of 0 to 100. These ratings are standardised and used in the implementation to check the metric is performing adequately. For both French and German, there are 1000 reference translations in the dataset. Each reference has one or more candidate translations, giving a total of 2521 translations for French and 3485 for German. While there were 25 assessors for German there were only 7 for French, which may result in some variance [20]. The results from these French and German experiments are largely preliminary.

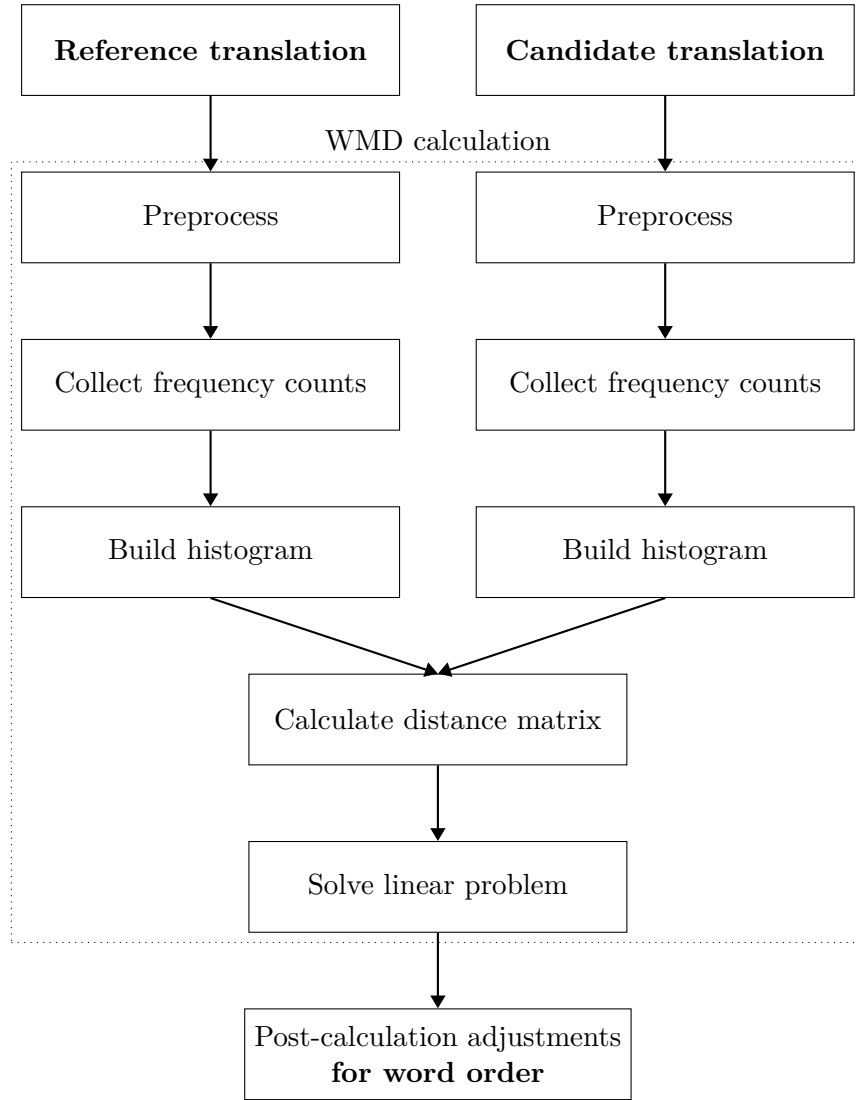


Figure 3.2: Flow of metric evaluations.

English Data

The data from the Metrics evaluation task also includes scores of other metrics alongside the human scores, meaning a direct comparison of correlation coefficients can be used to rate one metric over another. These metrics include traditional syntactic methods in BLEU as well as more modern methods such as BLEND which combine many previous metrics [6]. The human scores are also given 15 repeat assessments, the minimum to achieve reliable results [25]. These were combined into a single mean score. For the data from the metrics task, each language pair has results for 560 different translations. This is sampled randomly to avoid selecting identical sentences which can impact the human annotator’s agreement and consistency.

To better aid the experimental procedure, the data provided was formatted into JSON; for each translation language-pair there would be one file for the reference translations and one for the candidate translations, linked by an ID provided in the dataset.

```
{
  "ID": "Translation",
  :
}
```

Listing 3.1: Formatted reference translations

```
{
  {
    "ID": [
      [
        "Translation",
        Score
      ],
      :
    ],
    :
  }
}
```

Listing 3.2: Formatted candidate translations

3.1.2 Preprocessing

To ensure each translation is of the same format before any work is done, ample preprocessing is required to transform every sentence into an easily tokenisable result. Each character in the candidate and reference translations are converted into lower case, avoiding scenarios where “*Hello*” and “*hello*” would not match on processing. Punctuation is also stripped from the string where appropriate, but separation is preserved. For example the phrase “*29-year-old*” is separated to “*29*”, “*year*”, and “*old*”. This does mean that some implied meanings may be ignored; a forward slash in “*A/B*” can refer to the alternative of “*A*” or “*B*”, but can also be used to signify a mathematical ratio in “*km/h*”. Replacing the slash with a space defaults this to the former case, but this is where the majority of slashes in a text come from. Apostrophes are also a tricky edge case; transforming “*don’t*” to “*do*” and “*n’t*” works just as well as preprocessing it to “*dont*” as word embeddings exist for both cases, the latter style is used for simplicity. Percentage and hash symbols are kept as they have semantic value, but numerical values with decimal points and commas have these removed. Popular phrases that contain punctuation also remove punctuation for consistency, so the abbreviation “*N.Y.*” becomes “*ny*”.

This is an example output of a sentence being preprocessed, where whitespace becomes the delimiter for tokenisation:

```
Original sentence: This 3 dollars is $3, 10% of my income/thirty-percent of my
↪ future. #work
Preprocessed sentence: this 3 dollars is $ 3 10 % of my income thirty percent of
↪ my future # work}
```

3.2 Word Mover's Distance

Word Mover's Distance is key to this implementation as it provides a useful vector space metric for embeddings. To calculate the WMD between a pair of sentences, the vectors and frequency counts of each word are used to generate normalised bag-of-words vectors for each sentence, which can be used to calculate the word “travelling” distance between all the tokens. Using the PyEMD library [41] [49] [50], this distance matrix can be used along with the two sentences' word occurrence histograms to get a WMD value. The result also provides a flow matrix, illustrating which words in the candidate translation have been mapped to each word in the reference translation; the flow of the transportation problem.

```
wmd(ref, cand, wordvectors):
    ref_list = tokenise(ref)
    cand_list = tokenise(cand)

    words = dictionary(ref_list + cand_list) #learning vocabulary dictionary
    v_ref, v_cand = frequency([ref, cand]) #transforming to token counts
    v_ref /= v_ref.sum() #building occurrence histogram
    v_cand /= v_cand.sum()

    wvs = []
    for word in words: #collecting all word vectors
        wvs.append(wordvectors[word])

    distance_matrix = distance(wvs) #distance metric to create matrix

    wmd, flow = emd(v_ref, v_cand, distance_matrix) #solution of linear program

    return wmd, flow
```

Listing 3.3: Pseudocode for getting the Word Mover's Distance

The resulting WMD value is a non-negative number, while the flow matrix gives an indication of the proportions of each word getting mapped to another. Given the following two sentences:

Reference: Bright glowing skies on sunny days.

Candidate: Clear bright blue sky today.

the following token lists and vocabulary dictionary are obtained:

```
ref_list: ['bright', 'glowing', 'skies', 'on', 'sunny', 'days']
cap_list: ['clear', 'bright', 'blue', 'sky', 'today']
words: ['blue', 'bright', 'clear', 'days', 'glowing', 'on', 'skies', 'sky',
↪ 'sunny', 'today']
```

The vocabulary dictionary gives an index to each word that occurs, ordered alphabetically. This can be used to set up the histograms as well as the distance matrix. The distance matrix is symmetric, with each value representing the vector distance between the embeddings of each respective word. Each row and column of the matrix represents a word, also ordered alphabetically.

```

v_ref:      [0.  0.17 0.  0.17 0.17 0.17 0.17 0.  0.17 0. ]
v_cand:     [0.2 0.2  0.2 0.   0.   0.   0.   0.2 0.   0.2]
distance_matrix: [[0.   0.42 0.58 0.68 0.53 0.62 0.49 0.38 0.59 0.68]
                  [0.42 0.   0.45 0.65 0.39 0.64 0.53 0.47 0.35 0.65]
                  [0.58 0.45 0.   0.68 0.56 0.68 0.61 0.63 0.58 0.62]
                  [0.68 0.65 0.68 0.   0.75 0.66 0.56 0.63 0.58 0.53]
                  [0.53 0.39 0.56 0.75 0.   0.68 0.63 0.59 0.53 0.75]
                  [0.62 0.64 0.68 0.66 0.68 0.   0.73 0.67 0.68 0.52]
                  [0.49 0.53 0.61 0.56 0.63 0.73 0.   0.24 0.47 0.64]
                  [0.38 0.47 0.63 0.63 0.59 0.67 0.24 0.   0.53 0.67]
                  [0.59 0.35 0.58 0.58 0.53 0.68 0.47 0.53 0.   0.67]
                  [0.68 0.65 0.62 0.53 0.75 0.52 0.64 0.67 0.67 0.  ]]

```

Listing 3.4: Histograms and distance matrix

From this, the linear program can be solved. This returns a WMD value as well as a flow matrix detailing which words transform to which other words in the opposite sentence. Transforming the reference translation to the candidate translation is essentially the same problem as transforming the candidate translation; the implementation will use the former. In the flow it can be seen how much

```

wmd:      0.405
flow:     [[0.   0.   0.   0.   0.   0.   0.   0.   0.   0. ]
            [0.   0.17 0.   0.   0.   0.   0.   0.   0.   0. ]
            [0.   0.   0.   0.   0.   0.   0.   0.   0.   0. ]
            [0.   0.   0.   0.   0.   0.   0.   0.   0.   0.17]
            [0.07 0.   0.1  0.   0.   0.   0.   0.   0.   0. ]
            [0.13 0.   0.   0.   0.   0.   0.   0.   0.   0.03]
            [0.   0.   0.   0.   0.   0.   0.   0.17 0.   0. ]
            [0.   0.   0.   0.   0.   0.   0.   0.   0.   0. ]
            [0.   0.03 0.1  0.   0.   0.   0.   0.03 0.   0. ]
            [0.   0.   0.   0.   0.   0.   0.   0.   0.   0.  ]]

```

Listing 3.5: WMD and flow result

of each word from the reference translation “distribution” gets moved to the candidate translation equivalent, which is why words which do not appear in the reference translation have a row of all zeroes. All the entries in the matrix will add up to 1, with each word having equal weighting. In this case, each word would tally up to one-sixth, with the word “glowing” in the sixth row mapping four-fifths to “blue” and one-fifth to “today”, for example.

3.3 Word Embeddings

As the focus of the implementation is mostly on the metric itself, the word embeddings used are largely pre-trained embeddings. These are taken from different available sources online. While this does mean the different training parameters and datasets gives each embedding a different level of quality, the results from each can still be used to develop the distance algorithms as the embeddings remain a controlled variable.

In these experiments, word embeddings for German, French, and several different English em-

beddings were used. These different embeddings range in quality and function, as some are simple word2vec embeddings while others are FastText embeddings, which have the benefit of using n-grams to build out-of-vocabulary words. While the German and French embeddings were from fastText, they were used as word2vec embeddings for simplified loading times. All embeddings used were of dimension 300 for consistency and detailed in Table 3.1.

Language	Type	Vocabulary size	Dimension	Source
German	word2vec	2000000	300	[GermanEmbedding]
French	word2vec	2000000	300	[21]
English	word2vec	3000000	300	[65]
English	word2vec (trained)	773377	300	[65]
English	fastText	2000000	300	[43]
English	fastText (trained)	1066155	300	[43]

Table 3.1: Word embeddings used in the experiments

3.4 Tweaking WMD

Some parameters within the WMD formula were open to tuning, such as the distance function used, the handling of words not within the embedding, and the use of stop words. The following section looks at several experiments to find the best parameters for the standard WMD.

3.4.1 Distance measure

Starting off with the German and French data, the first experiment was to find the best performing distance metric for the WMD calculation. While the original WMD uses Euclidean distance to determine the distance between words in the word2vec space, the Cosine distance is also a viable option as a distance measure as it is less affected by vector magnitude, potentially working better with the higher dimension vectors used in most word embeddings.

Calculating WMD with Euclidean and Cosine distance measures on the two language datasets, the resulting Pearson and Spearman correlations are found in Table 3.2.

	Pearson		Spearman	
	$ r $		$ r_s $	
	en-de	en-fr	en-de	en-fr
Euclidean	0.543	0.500	0.546	0.489
Cosine	0.556	0.526	0.562	0.515

Table 3.2: Absolute Pearson and Spearman correlation of German and French data using both Euclidean and Cosine distance as its distance metric.

From these results it shows that the Cosine distance improved the correlation with human scores for both languages. This is visualised in Figure 3.3 and 3.4.

Notably, there are a few vertical lines on the scatter plots, which shows that many of the human scores are in fact the same, a weakness of the dataset given. Nevertheless, the relationship between all the assessed sentence pairs is quite linear, explaining the strong correlation scores and the slope on the scatter diagram.

However, one issue that plagued both of these language pairs was the issue of missing words. The German set had an out-of-vocabulary rate of 0.3% and the French 0.6%; not high values but still significantly in the hundreds/thousands of missing words given the size of the vocabulary. For these

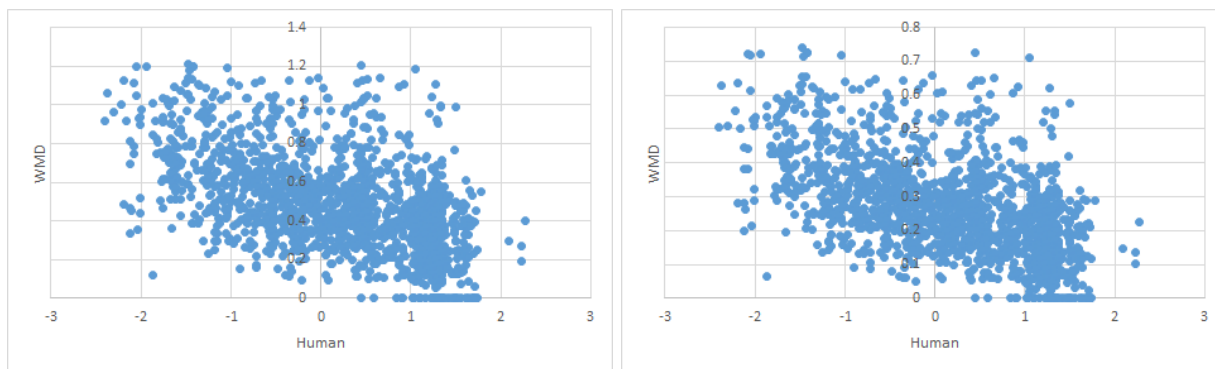


Figure 3.3: WMD against Human scores for German, using Euclidean (left) and Cosine (right) distance.

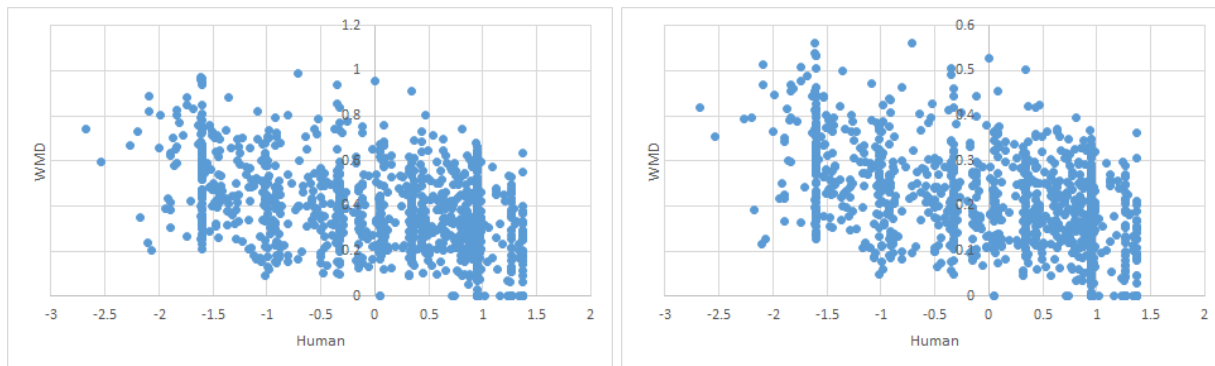


Figure 3.4: WMD against Human scores for French, using Euclidean (left) and Cosine (right) distance.

experiments, sentence pairs with any missing words were excluded from calculations, which amounted to 1712 out of 3247 in German and 1291 out of 2520 for French, a very large number. Due to my own inexperience in these languages, this became quite a difficult issue to debug.

As a result of these missing words as well as the general lack of high quality embeddings and datasets available, the rest of the experiments were done using English embeddings and datasets.

3.4.2 Missing words

Using the English word2vec embedding and the Cosine distance for WMD, there were improvements in the out-of-vocabulary rate to 0.1% as a result of the larger vocabulary. Most sentence pairs contained missing words, making it nearly impossible to disregard them in this case. However, these human ratings are much more varied and avoid the vertical line phenomenon found in the German and French set.

A strategy for handling missing words consistently was therefore necessary to deal with any encountered sentence pairing, even if overall correlation suffers. This should assign a vector within the semantic space to the missing word, retrieving the same vector when the missing word is come across again. This allows it to be treated like any other word within the vocabulary. In code, this is best performed with the Python dictionary as illustrated in Listing 3.6. Several strategies for achieving this are viable – this section looks to find the best performing one across the seven language pairings into English.

Random vector

One approach to assigning vectors to missing words is to have a random vector for each. This finds the maximum and minimum value for each dimension within the original embedding and selects a


```

wmd(ref, cand, wordvectors, missing):
    :
    wvs = []
    for word in words: #collecting all word vectors
        if word in wordvectors:
            wvs.append(wordvectors[word])
        else:
            if word not in missing:
                missing[w] = generate_vector(w)
            wvs.append(missing[w])
    :

```

Listing 3.6: Pseudocode for storing and retrieving missing word vectors

random value between those two numbers for each dimension of the missing word vector.

Single vector

Another way of assigning vectors is to have a single “out-of-vocabulary” vector for all missing words. One such vector could be the zero vector, as it is likely the most neutral of all values. A random vector could also be used for each iteration but this would create too much variation for analysis.

Average vector

Taking the average of several different vectors to create the missing word vector is also a possible method. For these experiments, the average of five random vectors in the vocabulary was used for each word.

fastText

A different approach to assigning a new vector to each missing word is to utilise fastText embeddings. Building word vectors of missing words by aggregating the vectors of its n-grams, the fastText embeddings will ensure there are no out-of-vocabulary words. However, the performance may be different against that of word2vec as the embeddings are also of different quality. The embedding will also be used as a word2vec embedding, without n-gram functionality, to test if performance change is a result of vector quality rather than function. This was tested using the best performing technique on word2vec.

Trained embeddings

Another way to try different embeddings is to train them specifically with the dataset that the test data is based on. Using the given English corpus [40], a word2vec embedding and a fastText embedding was trained. Ideally this would help pick up missing words from the test dataset as they come from the same context. Both models used the skip-gram model, with a minimum count of 5, window size of 10, and a negative sampling “noise words” value of 5 for consistency. These parameters were largely based on previous empirical studies [36] [44].

Performance

Assessing the different approaches on the English dataset, it shows that some methods work better than others across the language pairs, seen in Table 3.3.

Embedding	Method	cs-en	de-en	fi-en	lv-en	ru-en	tr-en	zh-en	OOV (%)
word2vec	Random vector	0.513	0.531	0.687	0.501	0.560	0.557	0.591	0.10
word2vec	Single vector (zero)	0.513	0.531	0.689	0.505	0.562	0.561	0.595	0.10
word2vec	Average vector	0.500	0.534	0.678	0.492	0.563	0.557	0.572	0.10
fastText	Using n-grams	0.511	0.542	0.700	0.526	0.572	0.577	0.583	0
word2vec (tr.)	Single vector (zero)	0.494	0.527	0.685	0.520	0.546	0.539	0.603	0.24
fastText	Single vector (zero)	0.521	0.536	0.704	0.530	0.571	0.566	0.607	0.22
fastText (tr.)	Single vector (zero)	0.485	0.525	0.671	0.513	0.546	0.538	0.597	0.18

Table 3.3: Absolute Pearson correlation of English data for different methods of resolving missing words. The bottom half of the table shows results of different embeddings using the zero vector method.

Of the word2vec methods, the single zero vector performs the best, although having a random vector for each missing word is only slightly worse. The n-gram method of fastText performs better than these, but this seems more to be an effect of the word embedding quality itself. When the fastText embedding is used with the single zero vector method instead, it has higher correlations with most language pairs. While there is not much to split the two, the better performing fastText embedding with zero vector for missing words is used going forward with these experiments.

The trained embeddings had fairly disappointing results and out-of-vocabulary handling, which can be traced to several reasons. The first is the minimum count parameter being set to 5, which was necessary to preserve vector quality, but meant that genuine rare words were not encapsulated. The second is the training set itself was not as big as the pre-trained one, resulting in a vocabulary size far smaller. Perhaps most significantly, a large majority of the missing words found were not even English words, but words from the source language which hadn’t been translated properly. The excerpt in Listing 3.7 gives an indication of this.

```

:
analysoiman
pārraugošo
kurzeme
deder
landzman
:

```

Listing 3.7: Excerpt of the missing words found in dataset

3.4.3 Stop words

A deliberation which had to be made when developing WMD was whether to remove stop words in the calculation. Stop words in natural language processing are generally prepositions or “function words”, which do not add much to the content of the sentence but are required to keep it grammatically sound. While the original WMD does not include stopwords in their calculations, their purposes were more for finding general document similarity. When it comes to assessing translation quality on a

segment-level basis, including stopwords is more sensible as every element of the sentence is integral to its translation. This is particularly the case when looking at fluency as each word should be in the right order for it to be considered a strong translation. Unfortunately not every function word is in a pre-trained embedding as a result of how commonly they are ignored in language processing. Nevertheless, with the generic zero vector applied, including stop words in the calculation still performs better than removing all stop words before calculating WMD, as seen in Table 3.4.

	cs-en	de-en	fi-en	lv-en	ru-en	tr-en	zh-en
All words included	0.521	0.536	0.704	0.530	0.571	0.566	0.607
Stop words removed	0.491	0.524	0.677	0.524	0.546	0.558	0.601

Table 3.4: Absolute Pearson correlation of English data with and without stop words in the WMD calculation.

3.5 Baseline WMD results

Taking the optimal parameters from the previous sections, the baseline WMD scores for the English language pairs are given in Table 3.5. This uses the Cosine distance on the pre-trained fastText embedding, resolving missing words to the zero vector and including all words, including stopwords.

	cs-en	de-en	fi-en	lv-en	ru-en	tr-en	zh-en
WMD	0.521	0.536	0.704	0.530	0.571	0.566	0.607

Table 3.5: Absolute Pearson correlation of baseline WMD on English data.

The scatter diagrams for these correlations are shown in Figure 3.6.

From the graphs it is fairly evident each language pair has a negative correlation between WMD and Human scores. Every graph has a fairly concentrated cluster of points along this correlation, particularly in the middle. Where the line starts to fray is nearer the two ends, where the human scores for a translation are high or low. This has introduces a lot more volatility in the WMD scores, ranging very differently from its human counterpart.

This suggests that the standard WMD is not able to deal with translations which are extremely good or extremely bad in the same way they can handle a regular translation. WMD on its own essentially only handles the adequacy of words in a translation; there is no fluency component, which can compromise the overall evaluation of a sentence. Figure 3.5 gives an example of this. The sentence gets a perfect WMD score because all of its words align exactly to another one in the vector space, despite the words being in the complete opposite order. It is necessary to make distinction of fluency through some sort of measure. This fluency element could be a factor taking a translation from good to very good or bad to very bad, and is in line with the multifaceted human approach to translation evaluation which recognises both adequacy and fluency. Adding this to the WMD calculation is key to the next few sections.

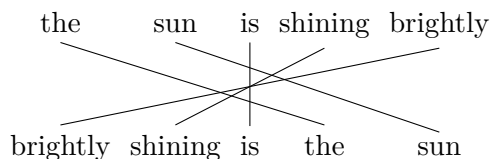
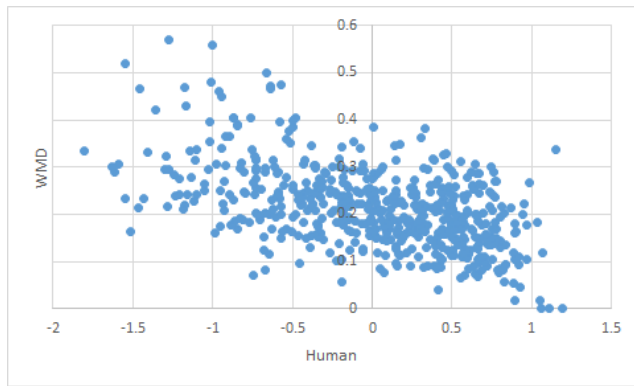
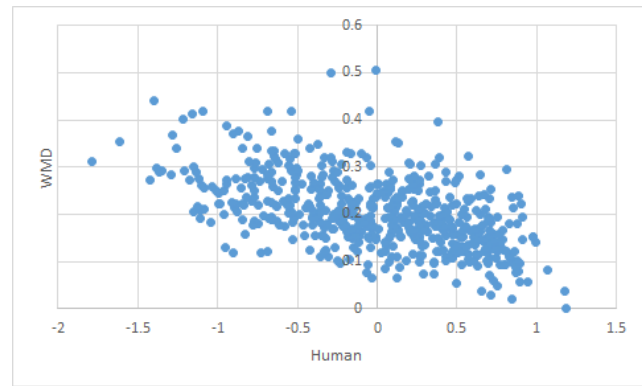


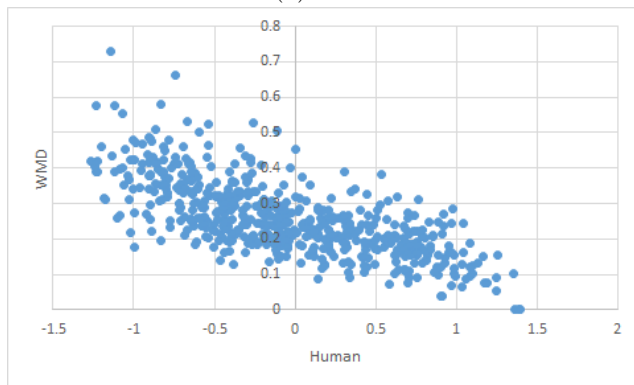
Figure 3.5: The WMD score for this sentence pair is 0.0 despite the words being in a completely different order.



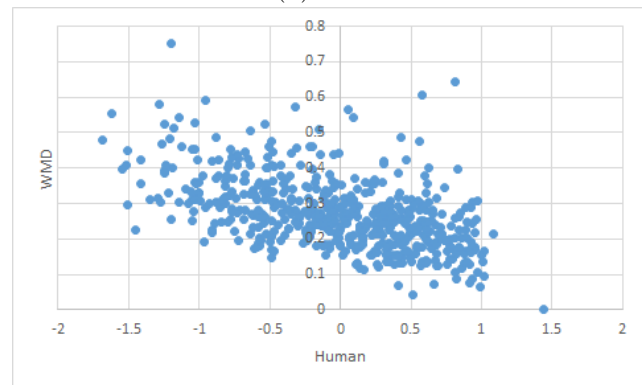
(a) cs-en



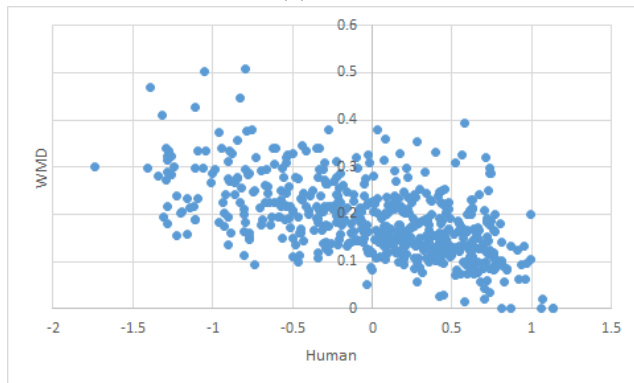
(b) de-en



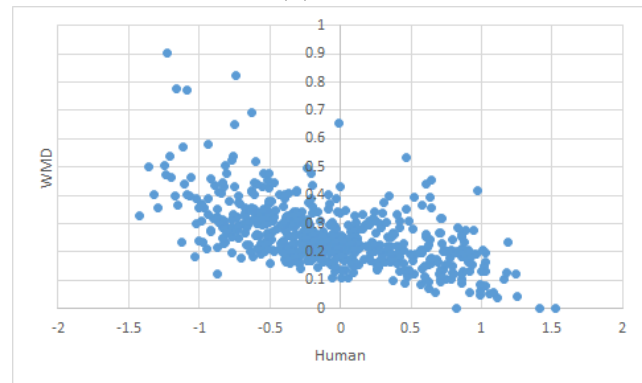
(c) fi-en



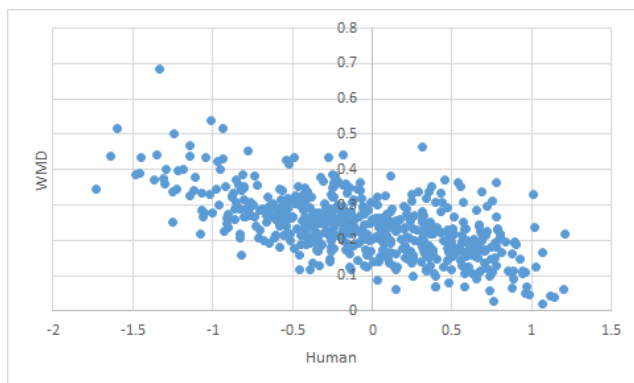
(d) lv-en



(e) ru-en



(f) tr-en



(g) zh-en

Figure 3.6: WMD against Human scores for the seven language pairs.

3.6 Word order penalties

The most basic approach to calculating fluency is to add a penalty on every time a matched word in the candidate sentence is not in the same position as its counterpart from the original reference. Implementation can be done using the WMD flow matrix as it provides an indication of which words in the reference map to those in the candidate translation. This is added to the result of WMD.

As sentence pairs are naturally not the same length, the method used to match the words was to find each word's relative position in the sentence, valued between 0 and 1. This would naturally reward sentences of the same length, while adding a slight penalty to each word if the sentences were of different sizes. Figure 3.7 gives an illustration of this.

Index	0	1	2	3	4	5	6
	Today	is	an	important	day	for	us
Relative Position	0	1/6	2/6	3/6	4/6	5/6	1

Figure 3.7: Example of a relative word position calculation

The code within Listing 3.8 walks through the algorithm used. Going through the flow matrix, which maps words from the reference to the candidate sentence, the relative position of each word in the reference is checked against the relative position of each word it matches to in the candidate to build up this penalty value. An interesting edge case is where two of the same word appears in the reference; to solve this the helper function `closest` picks the case with the closest index pair, reflecting the calculations of WMD. If a sentence only has one word, the word is given a position of 0.5, to reflect its position as the first and last word of the sequence. Figure 3.8 and 3.9 give two simple examples of this penalty being applied.

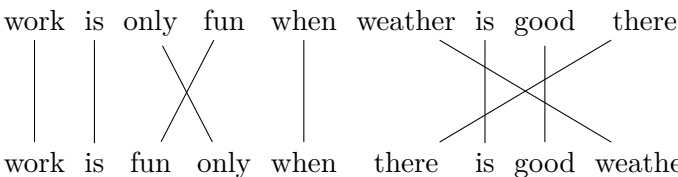
```

penalty(ref_list, cand_list, flow, words):
    penalty = 0
    for idx_ref, val_ref in enumerate(flow):
        word_weight = sum(val_ref)
        for idx_cand, val_cand in enumerate(val_ref):
            if val_cand != 0:
                word_ref = words[index_ref]
                word_cand = words[index_cand]
                pos_refs = relative_position(word_ref, ref_list)
                pos_cands = relative_position(word_cand, cand_list)
                proportion = val_cand/wordweight

                for pos_ref in pos_refs:
                    #finds closest index of word in candidate sentence
                    mapped_pos_cand = closest(pos_cands, pos_ref)
                    word_penalty = abs(pos_ref - mapped_pos_cand) * proportion
                    penalty += word_penalty
    return penalty

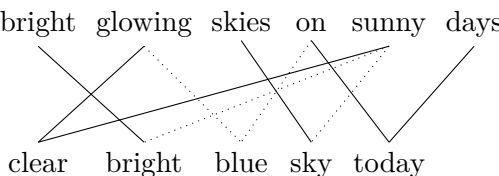
```

Listing 3.8: Pseudocode for calculating a word order penalty



Word	Reference position	Matched position	Matched word	Word penalty
work	0	0	work	0
is	1/8	1/8	is	0
only	2/8	3/8	only	1/8
fun	3/8	2/8	fun	1/8
when	4/8	4/8	when	0
weather	5/8	1	weather	3/8
good	7/8	7/8	good	0
there	1	5/8	there	3/8

Figure 3.8: The total penalty for this example would be 1. Note that “*is*” appears twice in the reference and the closest pairing is used.



Word	Ref. position	Match position	Match word	Proportion	Penalty	Word penalty
bright	0	1/4	bright	1	0.25	0.25
glowing	1/5	2/4	blue	2/5	0.12	0.24
glowing	1/5	0	clear	3/5	0.12	
skies	2/5	3/4	sky	1	0.35	0.35
on	3/5	2/4	blue	4/5	0.08	0.16
on	3/5	1	today	1/5	0.08	
sunny	4/5	1/4	bright	1/5	0.11	
sunny	4/5	0	clear	3/5	0.12	0.24
sunny	4/5	3/4	sky	1/5	0.01	
days	1	1	today	1	0	0

Figure 3.9: The total penalty for this example would be 1.24.

The results from applying these penalties can be seen in Table 3.6 and Figure 3.10. The performance of this was much poorer than the baseline WMD. Upon analysis of the graphs, it showed that the penalty was punishing sentences too strongly, particularly those which had many words out of position. This compounded the increase in metric score such that they were far removed from the main cluster.



Figure 3.10: WMD with word order penalty against Human scores for the seven language pairs.

	cs-en	de-en	fi-en	lv-en	ru-en	tr-en	zh-en
WMD	0.521	0.536	0.704	0.530	0.571	0.566	0.607
WMD word order penalty	0.265	0.261	0.391	0.270	0.315	0.319	0.331

Table 3.6: Absolute Pearson correlation of baseline WMD on English data.

A method of handling this growth is to cap the penalty at a certain value. This allows the penalty to grow at the same rate, but anything above a certain threshold would all be penalised by a given maximum value. Experiments were run with the maximum value at 0.05, 0.1, 0.15, 0.2, 0.25, 0.5, 0.75, 1, 5, 10 and 15. The results suggest that any additional penalty to the baseline WMD must be fairly small, as penalties that skyrocket create too much of a disparity between values, weakening correlation. This is illustrated in Figure 3.11. As shown in Table 3.7, the best performing maximum value was the one closest to 0. However, capping the penalty at such a low value essentially gives every contentious sentence the same penalty, making it hard to differentiate between a sentence that only has a slight loss of fluency against a sentence with every word in a different position. In addition, the best performing penalties are also barely outperforming the baseline WMD, suggesting this algorithm for fluency is not particularly useful and practically negligible.

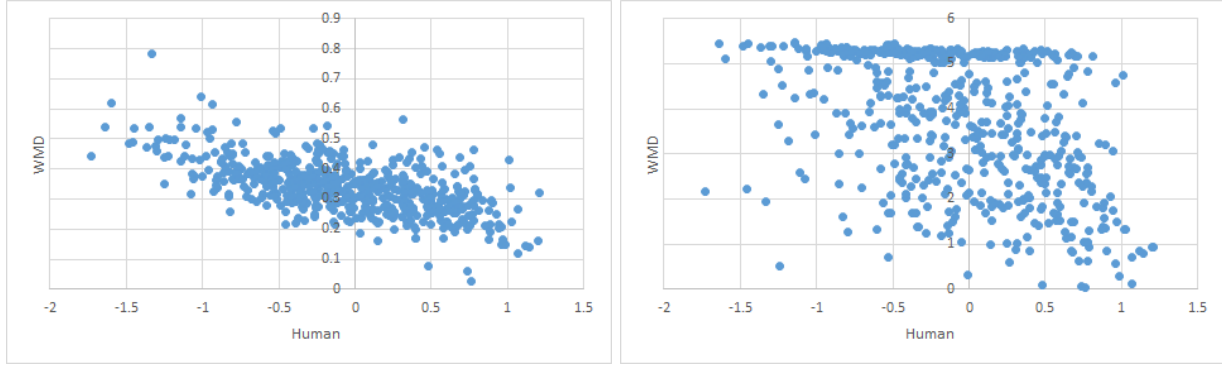


Figure 3.11: WMD with word order penalty against Human scores for Czech data, with maximum at 0.1 (left) and 5 (right).

	cs-en	de-en	fi-en	lv-en	ru-en	tr-en	zh-en
WMD	0.521	0.536	0.704	0.530	0.571	0.566	0.607
WMD word order penalty	0.265	0.261	0.391	0.270	0.315	0.319	0.331
WMD word order penalty, max 0.05	0.525	0.535	0.703	0.531	0.569	0.567	0.607
WMD word order penalty, max 0.1	0.524	0.529	0.697	0.531	0.561	0.567	0.604
WMD word order penalty, max 0.15	0.519	0.518	0.688	0.530	0.549	0.566	0.600
WMD word order penalty, max 0.2	0.511	0.503	0.676	0.528	0.533	0.564	0.595
WMD word order penalty, max 0.25	0.500	0.487	0.662	0.525	0.516	0.561	0.588
WMD word order penalty, max 0.5	0.440	0.406	0.583	0.501	0.435	0.539	0.541
WMD word order penalty, max 0.75	0.396	0.358	0.521	0.467	0.375	0.522	0.490
WMD word order penalty, max 1	0.367	0.323	0.473	0.425	0.336	0.506	0.465
WMD word order penalty, max 5	0.309	0.303	0.407	0.311	0.325	0.385	0.433
WMD word order penalty, max 10	0.266	0.263	0.396	0.282	0.316	0.330	0.360
WMD word order penalty, max 15	0.265	0.261	0.391	0.270	0.315	0.320	0.333

Table 3.7: Absolute Pearson correlation of baseline WMD on English data.

The discrepancy between penalties before the value is capped is largely because the algorithm compounds consecutive words which are out of order. The example in Figure 3.12 illustrates this.

Even though the words “the sun and “*shining brightly on a cool summer day*” are in the same order in both sentences, each word is still penalised for being out of position. Despite many of the words remaining in the same relative order, this has a penalty score of 4.67, even higher than the jumbled, nonsensical sentence in Figure 3.13 which scores 3.78.

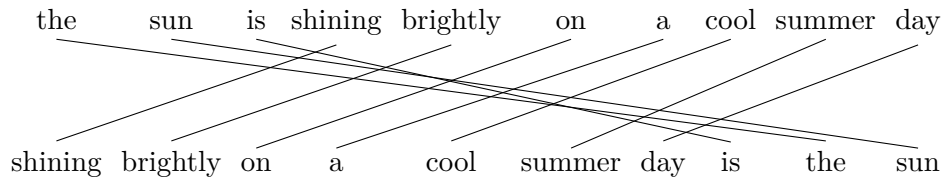


Figure 3.12: This sentence pair has a WMD score of 0.0 and a penalty of 4.67, despite many of the subsequences being the same.

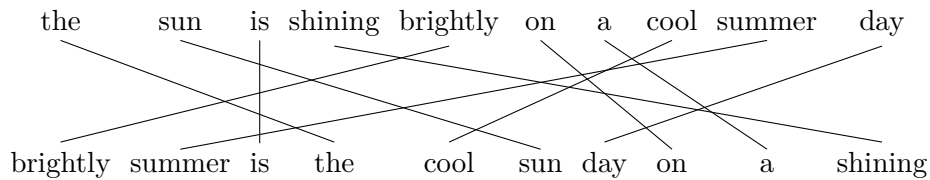


Figure 3.13: This sentence pair has a WMD score of 0.0 but only a penalty of 3.78, even though the words are heavily shuffled and the sentence is far more nonsensical.

3.7 Fragmentation penalties

To prevent the metric from doubly-penalising consecutive out of order words in the translation, the word order penalty can make use of fragments rather than individual words to achieve a fluency measure. This is similar to the fragmentation penalty of METEOR [1], which separates word matches into chunks. To refresh, chunks are a group of unigrams which are adjacent in both the reference and the candidate translation. The longer each chain of n-grams is, the fewer chunks there are, meaning if the entire candidate matches the reference there is just one single chunk. The ratio between chunks and matched unigrams is the key to this fluency component.

Re-examining the same sentence pairs from Figure 3.12 and 3.13, the former can be seen to be much less fragmented than the latter, and penalised accordingly. This is visualised in Figure 3.14 and 3.15. It also helps to penalise more nonsensical translations, as the fluency of the reference translation is more accurately reflected in the level of fragmentation.

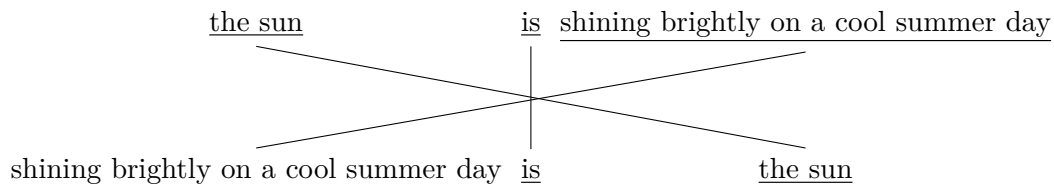


Figure 3.14: This sentence pair has 3 chunks and 10 matched unigrams, giving a ratio of 0.3.

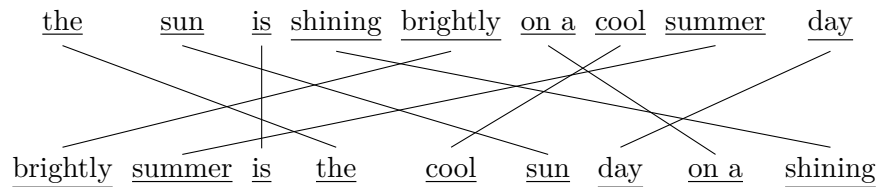


Figure 3.15: This sentence pair has 9 chunks and 10 matched unigrams, giving a ratio of 0.9.

Listing 3.9 gives an indication of how the fragmentation is calculated. For this implementation, there is assumed a one to one mapping of words from the reference to the candidate translation. In other words, each word in the reference is only considered to be mapped to one word in the candidate translation, even if the flow matrix moves it to multiple words. This means that only the most significant set of chunks are considered for the fragmentation calculation. This is because fragmentation is a sentence level construct, while the previous word order penalty could be applied on an individual word basis.

```

penalty(ref_list, cand_list, flow, words):
    chunks = 1
    matched_unigrams = 0

    current = -1
    for i, w in enumerate(ref_list):
        #finds which word/s the reference word maps to most
        index = words.index(w)
        flow_from_w = flow[index]
        highest_flow = max(flow_from_w)
        highest_match_indexes = [i for i, x in enumerate(flow_from_w)
                                if x == highest_flow]
        highest_match_words = [words[i] for i in highest_match_indexes]

        #finds which word is closest to reference word
        matched_indices = []
        for m in matched_words:
            occurrences = []
            for i, x in enumerate(cand_list):
                if x == m:
                    occurrences.append(i)
            matched_indices.append(closest(occurrences, current))
        matched_index = closest(matched_indices, current)

        if not current + 1 == matched_index:
            chunks += 1
            current = matched_index
            matched_unigrams += 1

    return chunks / matched_unigrams

```

Listing 3.9: Pseudocode for calculating fragmentation in sentence

In METEOR, the fragmentation penalty takes the ratio between the number of chunks c and the number of matched unigrams u_m , raises it to the power of 3 and multiplies it by the maximum of 0.5 to scale. It is then applied by subtracting it from 1 and multiplying the original score by this new factor. To simplify the expression and to better tune these arbitrary parameters for the purposes of this metric, the “penalty” value referred henceforth is just the value of the ratio:

$$\text{Penalty} = \frac{c}{u_m}$$

Since the penalty value is a ratio, it is always restricted to be between 0 and 1. This means the final values will not balloon as much as the word order penalty did in the previous section, as that did not have a natural maximum. While restricting the maximum value of the word order penalty did help performance, it meant that many sentence pairs would end up with the same maximised penalty value. This ratio should allow a range of values between 0 and 1, allowing better distinction between sentence pairs.

3.7.1 Additive fragmentation penalty

This section will feature experiments to tune the values of the exponent n and maximum value δ , as well as tuning of its application to the baseline WMD value. Initially the experiments will just add the fragmentation component onto the base WMD score, similar to the previous section. This gives the following formula for the fragmentation penalty:

$$\text{WMD fragment} = \text{WMD} + \delta \times (\text{Penalty})^n$$

This results for this are in Table 3.8, compared against the baseline WMD and the best performing WMD word order penalty.

	δ	n	cs-en	de-en	fi-en	lv-en	ru-en	tr-en	zh-en
WMD	–	–	0.521	0.536	0.704	0.530	0.571	0.566	0.607
WMD word order	–	–	0.525	0.535	0.703	0.531	0.571	0.567	0.607
WMD fragment	0.1	1	0.531	0.546	0.710	0.541	0.585	0.600	0.621
WMD fragment	0.2	1	0.530	0.542	0.705	0.543	0.585	0.620	0.623
WMD fragment	0.3	1	0.525	0.534	0.696	0.540	0.580	0.631	0.621
WMD fragment	0.4	1	0.518	0.525	0.686	0.535	0.572	0.637	0.616
WMD fragment	0.5	1	0.511	0.516	0.676	0.529	0.564	0.640	0.611
WMD fragment	0.1	2	0.529	0.545	0.707	0.540	0.585	0.601	0.619
WMD fragment	0.2	2	0.524	0.540	0.699	0.538	0.585	0.617	0.618
WMD fragment	0.3	2	0.516	0.530	0.686	0.531	0.580	0.624	0.612
WMD fragment	0.4	2	0.508	0.520	0.674	0.523	0.573	0.626	0.606
WMD fragment	0.5	2	0.499	0.510	0.662	0.515	0.566	0.625	0.599
WMD fragment	0.1	3	0.525	0.543	0.704	0.537	0.582	0.596	0.615
WMD fragment	0.2	3	0.519	0.537	0.692	0.533	0.583	0.607	0.611
WMD fragment	0.3	3	0.509	0.527	0.677	0.524	0.578	0.610	0.603
WMD fragment	0.4	3	0.498	0.515	0.663	0.515	0.571	0.608	0.594
WMD fragment	0.5	3	0.488	0.505	0.649	0.506	0.564	0.605	0.586

Table 3.8: Absolute Pearson correlation of baseline WMD on English data.

Compared to the baseline WMD and the best performing word order penalty, the fragmentation penalty can be seen to have stronger correlation with human judgment. The effect of the exponentiation term n is fairly weak; its effect is mostly just pulling down the magnitude of the penalty, which leads

to less correlation with human scores. From the results it can be seen that leaving this value at $n = 1$ is the best approach. In terms of the maximum penalty value δ , the performance is best around lower values of 0.1 and 0.2, with the exception of the Turkish-English language pair, which has best performance at larger values of δ . This maximum value gives an indication of the optimal weight of a fragmentation penalty. Since the adequacy measure of baseline WMD with cosine distance is limited between 0 and 1, this result suggests that the fluency component is best suited to be around 10 to 20% of that.

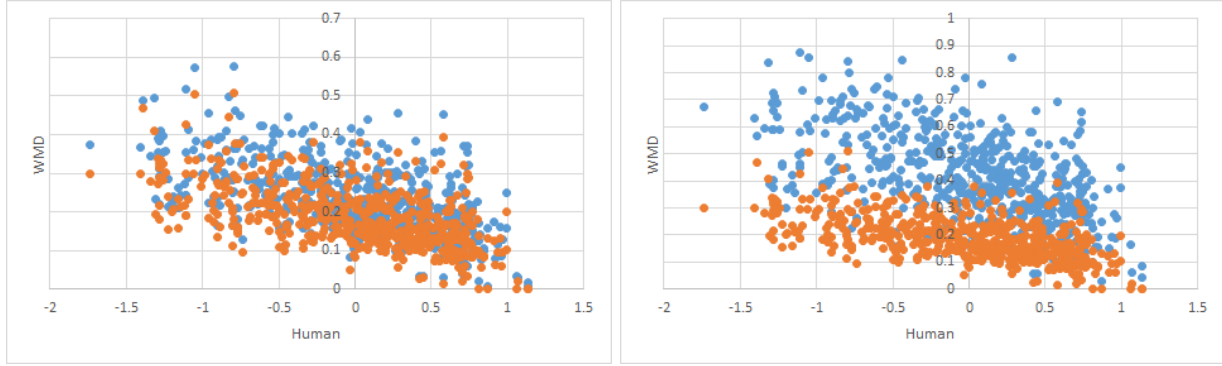


Figure 3.16: WMD against Human scores for Russian data. Orange is the baseline WMD, blue is with fragmentation penalty applied with $n = 1$ and δ at 0.1 (left) and 0.5 (right).

Analysing some of the results graphically in Figure 3.16, it can be suggested that correlation could be weakened because points are only travelling in one direction, upwards. This is compounded in the iterations with higher values of δ , as the sentences with extreme fragmentation get dragged further and further upwards while those without remain on the same footing. Another interesting take from the baseline WMD is its lower limit. Many of the graphs show several points along the horizontal axis where WMD is 0. Ideally the WMD score should only be at its lowest when the translation is at its best, but this is not the case.

3.7.2 Multiplicative fragmentation penalty

These issues could be addressed by allowing bidirectional movement of the WMD value, adjusting the fluency component up and down instead of just up. Sentence pairs can be rewarded for having high fluency or punished further for an non-fluent translation. In doing this, the metric should also be able to encompass a wider range in the fluency measure.

The original fragmentation penalty grows the fragmentation penalty of sentences linearly with the growth of their penalty ratio, so reward and punishment values cannot just be added and subtracted from the original WMD, as this would just be a constant defined by the ratio. This would give the exact same correlations when judged through the Pearson statistic, so multiplying the original WMD by a fragmentation factor is the preferred strategy. This gives the formula for the bidirectional fragmentation penalty as:

$$\text{WMD bid.-fragment} = \text{WMD} \times (1 - \text{Penalty} + \rho)$$

The newly introduced parameter ρ decides what proportion of the data is to be rewarded and punished, based on each sentence's penalty ratio. If the value of ρ is 0.3, all sentence pairs with less than 0.3 penalty ratio is multiplied by a factor less than 1, while anything with greater than 0.3 penalty ratio is multiplied by a factor greater than 1. This factor varies with the penalty ratio, so sentences with higher ratios are multiplied by a greater fragmentation penalty and sentences with lower ratios are multiplied by a lower fragmentation penalty. The parameter is tested at increments of 0.1, from 0.1 to 0.9, to find the best performing occurrence.

The results of the bidirectional fragment experiments are shown in Table 3.9 alongside the best performing metrics of previous sections.

	ρ	cs-en	de-en	fi-en	lv-en	ru-en	tr-en	zh-en
WMD	–	0.521	0.536	0.704	0.530	0.571	0.566	0.607
WMD word order	–	0.525	0.535	0.703	0.531	0.571	0.567	0.607
WMD fragment	–	0.531	0.546	0.710	0.543	0.585	0.640	0.623
WMD bid.-fragment	0.1	0.110	0.189	0.189	0.054	0.272	0.012	0.028
WMD bid.-fragment	0.2	0.200	0.264	0.326	0.160	0.337	0.112	0.157
WMD bid.-fragment	0.3	0.270	0.320	0.427	0.244	0.383	0.210	0.261
WMD bid.-fragment	0.4	0.323	0.363	0.499	0.306	0.418	0.282	0.339
WMD bid.-fragment	0.5	0.361	0.395	0.548	0.352	0.444	0.335	0.396
WMD bid.-fragment	0.6	0.390	0.419	0.583	0.386	0.464	0.374	0.437
WMD bid.-fragment	0.7	0.412	0.437	0.608	0.411	0.479	0.404	0.467
WMD bid.-fragment	0.8	0.428	0.452	0.626	0.429	0.491	0.426	0.490
WMD bid.-fragment	0.9	0.441	0.463	0.640	0.444	0.500	0.444	0.508

Table 3.9: Absolute Pearson correlation of WMD with bidirectional fragmentation penalty, $\rho < 1$ on English data.

The results of this alteration are not as strong as the basic fragmentation penalty. A reason for this could be how much the penalties and punishments are allowed to grow, creating a big disparity between values. It is notable that the best performance occurs when the value of ρ is closest to 1, suggesting the bidirectional element is not as useful as thought. Figure 3.17 gives an illustration of this. The graph with ρ at 0.1 has very poor correlation as the majority of sentences are multiplied by a factor close to 0, reducing them all to a very small range and almost creating a horizontal line. It is clear that to increase distinction between values any multiplicative factor will need to increase the value of WMD rather than decrease it.

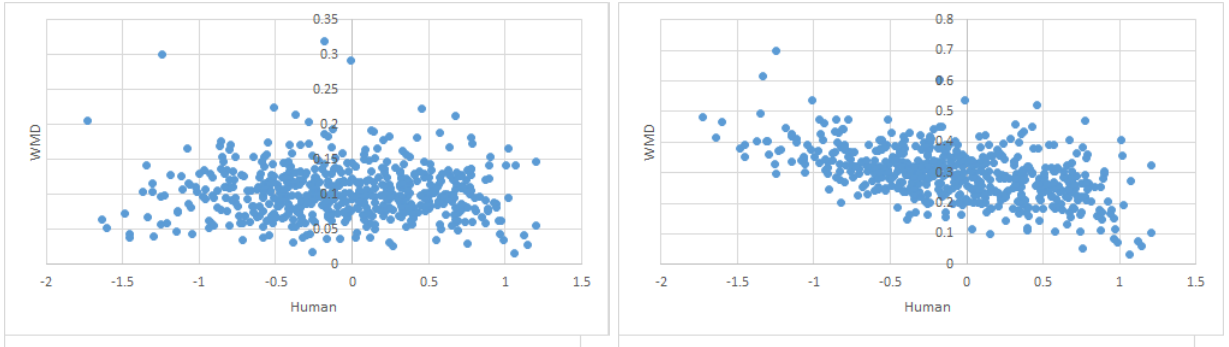


Figure 3.17: WMD against Human scores for Chinese data, with values of ρ at 0.1 (left) and 0.9 (right).

To tackle these issues another experiment was run with values of ρ at much higher values, the results of which are in Table 3.10. Performance reaches a plateau at high values of ρ , but still falls short compared to the basic fragmentation penalty.

The final implementation of this multiplied penalty takes the high ρ value of 10 and introduces the extra δ element to limit the size in which the multiplicative factor can grow.

$$\text{WMD bid.-fragment-limit} = \text{WMD} \times (1 - \delta \times (\text{Penalty} + \rho))$$

The results of this are in Table 3.11. While the results are improved from the non-limited version, it remains weak compared to the basic fragmentation penalty. These experiments show that introducing

	ρ	cs-en	de-en	fi-en	lv-en	ru-en	tr-en	zh-en
WMD	–	0.521	0.536	0.704	0.530	0.571	0.566	0.607
WMD word order	–	0.525	0.535	0.703	0.531	0.571	0.567	0.607
WMD fragment	–	0.531	0.546	0.710	0.543	0.585	0.640	0.623
WMD bid.-fragment	1	0.451	0.472	0.650	0.486	0.508	0.458	0.521
WMD bid.-fragment	1.25	0.469	0.488	0.668	0.497	0.522	0.483	0.544
WMD bid.-fragment	1.5	0.480	0.499	0.678	0.504	0.532	0.499	0.559
WMD bid.-fragment	1.75	0.488	0.506	0.684	0.509	0.538	0.510	0.568
WMD bid.-fragment	2	0.493	0.511	0.688	0.513	0.543	0.518	0.575
WMD bid.-fragment	2.5	0.500	0.517	0.694	0.518	0.550	0.529	0.583
WMD bid.-fragment	3	0.505	0.521	0.697	0.521	0.554	0.536	0.588
WMD bid.-fragment	3.5	0.508	0.524	0.699	0.523	0.557	0.541	0.592
WMD bid.-fragment	5	0.512	0.529	0.701	0.526	0.562	0.549	0.597
WMD bid.-fragment	7.5	0.516	0.532	0.703	0.528	0.565	0.555	0.601
WMD bid.-fragment	10	0.517	0.533	0.703	0.529	0.567	0.558	0.603

Table 3.10: Absolute Pearson correlation of WMD with bidirectional fragmentation penalty, $\rho \geq 1$ on English data.

a fluency component by a multiplicative factor is not as effective as adding it. In fact, even the best performing correlations are not too far off from the baseline WMD. As a result, the additive penalties are the ones used as the final version of the evaluation metric.

	δ	cs-en	de-en	fi-en	lv-en	ru-en	tr-en	zh-en
WMD	–	0.521	0.536	0.704	0.530	0.571	0.566	0.607
WMD word order	–	0.525	0.535	0.703	0.531	0.571	0.567	0.607
WMD fragment	–	0.531	0.546	0.710	0.543	0.585	0.640	0.623
WMD bid.-fragment	–	0.517	0.533	0.703	0.529	0.567	0.558	0.603
WMD bid.-fragment-limit	0.1	0.499	0.496	0.655	0.376	0.547	0.595	0.596
WMD bid.-fragment-limit	0.2	0.525	0.539	0.704	0.527	0.577	0.577	0.612
WMD bid.-fragment-limit	0.3	0.524	0.539	0.704	0.529	0.576	0.575	0.611
WMD bid.-fragment-limit	0.4	0.524	0.539	0.704	0.529	0.576	0.574	0.611
WMD bid.-fragment-limit	0.5	0.524	0.538	0.704	0.529	0.575	0.574	0.611

Table 3.11: Absolute Pearson correlation of WMD with limited bidirectional fragmentation penalty, $\rho = 10$ and varying δ , on English data.

A look at the correlation between each of the fragmentation penalty styles and the baseline WMD shows why the results of the basic additive fragmentation work much better. Table 3.12 shows that the multiplicative fragmentation penalty has a higher correlation with the baseline WMD as the additive fragmentation penalty, meaning the latter is better at separating itself from the baseline WMD and getting more varied, distinguishable results. This is again using the best performing variation of the metric as the comparison.

	cs-en	de-en	fi-en	lv-en	ru-en	tr-en	zh-en
WMD fragment	0.981	0.982	0.993	0.971	0.975	0.903	0.975
WMD bid.-fragment	0.997	0.997	0.998	0.994	0.997	0.998	0.998

Table 3.12: Absolute Pearson correlation of fragmentation penalty options with baseline WMD.

3.8 Tackling anomalies

Despite the fairly strong performance of the fragmentation penalty, there remained some standout anomalous results. This section looks to see if this can be addressed using some improvements to the basic fragmentation penalty.

3.8.1 Missing word penalty

One of the major causes of translations varying far from the mean is the issue of missing words. Although the out-of-vocabulary rate is not very high for the embeddings used, the dataset it was tested on had several sentences which were wholly mistranslated, causing it to be filled with indecipherable foreign words.

A notable example of this is in the Turkish-English dataset, which has the following sentence pair:

Reference: "THOSE WHO COMMITTED THE COUP ATTEMPT SHOULD BE CLEARED FROM THE STATE"
 Candidate: "Blow" GİRİŞİMİNDE BULUNANLARIN DEVLETEN TEMİZLENMESİ GEREKİYOR

The candidate translation is essentially nonsensical when viewed from an English context. Given the out-of-vocabulary resolution strategy of initialising a zero vector as the representation for each unique word, this type of occurrence where a large proportion of the machine translation is made of missing words will grow the fragmentation penalty, but not to the same scale that a human translator would. This is because the calculated fragmentation penalty still tries to fit a meaning to the word within the vector space framework, treating it in the same way as a sentence with all words in the vocabulary. On the other hand, a human translator is far more likely to treat this type of translation as nonsensical and give it a terrible rating.

This discrepancy should be reflected through a missing word penalty, adding on another variable to the baseline WMD and fragmentation penalty. This missing word penalty is based on the proportion of words in the candidate translation which are out of the vocabulary. This ratio is referred to as “Missing” is weighted by another parameter, α , which is experimentally adjusted in Table 3.13 against the best performing WMD fragment settings of each language pair from the previous section. The equation for this is as follows:

$$\text{WMD fragment-miss.} = \text{WMD} + \delta \times \text{Penalty} + \alpha \times \text{Missing}$$

The results showed that there was not a very large variation on the results between the different values of α . The best performing value for this parameter was 0.1, which was able to boost the scores of each language pair’s correlation, proving the effectiveness of this additional penalty. Of course, the gains were not spectacularly big, but are a good way to handle the rare cases that significantly weaken the correlation because of this out-of-vocabulary issue.

Nevertheless, one of the main issues that the missing word penalty is not able to handle cleanly is the case of proper nouns. Many of these names can be fairly obscure and not part of the word vector dictionary, marking it as a missing word and weakening the correlation.

3.8.2 Number vector

Another type of anomalous result seen in the results of the basic fragmentation penalty are the sentences which include numbers, many of which are not part of the vocabulary because of the enormity of variations in numbers. While the aforementioned missing word penalty can handle many of the cases involving actual words, missing numbers are a slightly different proposition, as these are known values with easily recognisable ties to other items in the vocabulary, namely other numbers.

	α	cs-en	de-en	fi-en	lv-en	ru-en	tr-en	zh-en
WMD	–	0.521	0.536	0.704	0.530	0.571	0.566	0.607
WMD word order	–	0.525	0.535	0.703	0.531	0.571	0.567	0.607
WMD fragment	–	0.531	0.546	0.710	0.543	0.585	0.640	0.623
WMD fragment-miss.	0.02	0.532	0.546	0.710	0.543	0.587	0.640	0.624
WMD fragment-miss.	0.04	0.532	0.546	0.711	0.543	0.588	0.640	0.624
WMD fragment-miss.	0.06	0.532	0.545	0.711	0.543	0.589	0.640	0.624
WMD fragment-miss.	0.08	0.532	0.545	0.712	0.542	0.589	0.640	0.624
WMD fragment-miss.	0.10	0.532	0.544	0.712	0.542	0.590	0.640	0.625
WMD fragment-miss.	0.12	0.532	0.544	0.712	0.542	0.591	0.641	0.625
WMD fragment-miss.	0.14	0.532	0.543	0.712	0.541	0.591	0.641	0.625
WMD fragment-miss.	0.16	0.532	0.542	0.711	0.541	0.592	0.640	0.624
WMD fragment-miss.	0.18	0.531	0.541	0.711	0.540	0.592	0.640	0.624
WMD fragment-miss.	0.20	0.531	0.540	0.711	0.539	0.592	0.640	0.624

Table 3.13: Absolute Pearson correlation of WMD with fragmentation penalty and missing word penalty on English data.

An example of this anomaly comes in the German-English language set:

Reference: They came in at 3: 51.942 on goal.

Candidate: They finished in 3:51.943.

The strategy used here to handle the specific case of numbers as an out-of-vocabulary word is to introduce a number vector, which is essentially an average vector of all instances in the vocabulary which are numbers. This replaces the zero vector as the vector representation for any out-of-vocabulary token which is a number. The comparison of results for this is shown in Table 3.14.

	cs-en	de-en	fi-en	lv-en	ru-en	tr-en	zh-en
WMD fragment-miss. (with n.v.)	0.531	0.545	0.711	0.543	0.592	0.640	0.624
WMD fragment-miss. (without n.v.)	0.532	0.546	0.712	0.543	0.592	0.641	0.625

Table 3.14: Absolute Pearson correlation of WMD with fragmentation penalty and missing word penalty, with and without the number vector on English data.

The results of this setting are effectively the same with and without the number vector. This is likely because of the rarity of these sentences and the fact that the number vector is an average of all numbers found. While this value may be slightly different from zero it may not be overly so despite their common denominator, as different numbers can carry different meanings. It would be too arbitrary and perhaps make the experiment ungeneralisable to assign the vector of one known number to all of the encountered out-of-vocabulary numbers, as there is no guarantee that the number in the candidate sentence hails from the same context and no guarantee that a chosen number is better suited than any other number. Indeed, even making this selection random would create too much uncertainty in the algorithm; as a result this number vector idea was not pursued further.

3.9 Finalising metric

To finalise the metric for comparison with other metrics in the dataset, the best performing parameters are selected for each language, as well as an overall best performing single parameter. This single value

gives a good starting point for any future language pairs which might use this metric before any tuning of the parameters.

The final metric of this project’s implementation is titled WMD_O , representing the ordering component of the algorithm that attests to its fluency. Complementing the base WMD’s adequacy implementation, the altered WMD adds a dimension of fluency in order to better match the multifaceted approach of translation evaluation. This is given as:

$$\text{WMD}_O = \text{WMD} + \delta \times \text{Penalty} + \alpha \times \text{Missing}$$

where Penalty is the ratio of chunks to matched unigrams, δ is the weight parameter for that fragmentation element, Missing is the ratio of missing words to total words in the candidate translation, and α the weight parameter for that missing word penalty element.

	cs-en	de-en	fi-en	lv-en	ru-en	tr-en	zh-en
δ	0.13	0.11	0.09	0.18	0.15	0.50	0.18
α	0.10	0.02	0.10	0.02	0.20	0.12	0.12

Table 3.15: Best performing values of δ and α for WMD_O .

The best values of δ and α for each individual language pair is given in Table 3.15. The best performing parameters overall are the single values of δ and α that give the best total value when summing up all the correlations of the seven language pairs. This gives 0.18 as the value of δ and 0.1 as the value of α . Most of the values for all the different language pairs are very similar, with the exception of Turkish-English. This is possibly to do with the sentence structure of the Turkish language as well as the sentences within the dataset provided.

	cs-en	de-en	fi-en	lv-en	ru-en	tr-en	zh-en
WMD_O	0.532	0.546	0.712	0.543	0.592	0.641	0.625

Table 3.16: Results of WMD_O with best performing δ and α .

The results of the metric on the seven language pair dataset is displayed in Table 3.16. While some anomalous results still remain, the majority of the points fit along fairly well on the negative correlation line, improved by the missing word penalty.

3.10 Alternative approaches not used

Some methods to introduce fluency to the metric were also initially considered but were abandoned for reasons of complexity and practicality.

The first was to include the position of a word in a sentence within the embedding itself, known as “position embeddings”. This would have allowed fluency to be included inside the WMD calculation itself, rather than applied as a post-calculation variable. However, these are mostly done with convolutional neural nets [23] or transformer-based architectures [61] for machine translation, making it less trivial to add to the code directly. Another idea was to change the calculation of WMD itself, introducing the fluency component during the solution of the linear problem. This was also a problem too complex to introduce in the timeframe of the project, so was left as a potential extension instead.

Part III

Conclusions

Chapter 4

Results

4.1 Performance against other metrics

Comparing the performance of the best performing WMD metric against the other metrics of the WMT17 task dataset, the results show that the metric performs at a competitive level, achieving respectable results against state-of-the-art automatic translation evaluation metrics. Taking the finalised WMD metric WMD_O , Table 4.1 records the performance of the developed metrics, with an entry for the best overall value of δ (0.18) as well as α (0.10). It also has an entry for an “ideal” value of δ and α , the combining the best performing values of the parameters for the different language pairs.

	cs-en	de-en	fi-en	lv-en	ru-en	tr-en	zh-en
AUTO DA	0.499	0.543	0.673	0.533	0.584	0.625	0.583
BEER	0.511	0.530	0.681	0.515	0.577	0.600	0.582
BLEND	0.594	0.571	0.733	0.577	0.622	0.671	0.661
BLEU2VEC_SEP	0.439	0.429	0.590	0.386	0.489	0.529	0.526
CHRF	0.514	0.531	0.671	0.525	0.599	0.607	0.591
CHRF++	0.523	0.534	0.678	0.520	0.588	0.614	0.593
MEANT_2.0	0.578	0.565	0.687	0.586	0.607	0.596	0.639
MEANT_2.0-NOSRL	0.566	0.564	0.682	0.573	0.591	0.582	0.630
NGRAM2VEC	0.436	0.435	0.582	0.383	0.490	0.538	0.520
SENTBLEU	0.435	0.432	0.571	0.393	0.484	0.538	0.512
TREEAGGREG	0.486	0.526	0.638	0.446	0.555	0.571	0.535
UHH_TSKM	0.507	0.479	0.600	0.394	0.465	0.478	0.477
WMD	0.521	0.536	0.704	0.530	0.571	0.566	0.607
WMD_O , $\delta = 0.18$, $\alpha = 0.10$	0.532	0.543	0.709	0.542	0.590	0.615	0.625
WMD_O , $\delta = \text{IDEAL}$, $\alpha = \text{IDEAL}$	0.532	0.546	0.712	0.543	0.592	0.641	0.625

Table 4.1: Performance of different metrics in the WMT17 shared task against the two proposed metrics. Trained/ensemble metrics are highlighted in grey. Bolded values signify the best performing non-trained metric for each language pair.

Looking at the correlation results given, WMD and WMD_O outperforms many of the established metrics. Notably, the BLEU-based metrics SENTBLEU, NGRAM2VEC and BLEU2VEC_SEP have much weaker scores, indicating the WMD metric’s success at outperforming traditional methods. Even the optimised latter two metrics, which make use of fuzzy matches on token and n-gram embedding similarities, are far worse than that of WMD. Of the metrics in the list, there are three which overall perform better than the WMD metrics: BLEND, MEANT_2.0 and MEANT_2.0-NOSRL. For the Russian-English language pair, the CHRF and CHRF++ are better performing, but is outdone by

WMD otherwise. This provides a vote of confidence for using semantic similarity over string similarity, as the approach is more fluid and allows the connotations of a text to shine through.

Of the metrics which outperform the implemented WMD, all are metrics which have evolved beyond the basic string matching approach. BLEND is a metric which combines many other metrics, and is also trained on data to produce a its strong scores [51]. This is less useful when applied to larger datasets, as practice of this metric requires fairly large amounts of human annotated data to train its model. Its position as an ensemble metric also means that there is a lot more complexity in calculation as it will depend on many different variables. The metrics that it uses include many classic lexical metrics such as BLEU and METEOR; this is complemented by more sophisticated metrics such as CHARACTER and BEER. The MEANT metrics, on the other hand, are not trained but able to achieve very high performance. This metric is similar to WMD as it uses distributional word vector models to evaluate semantic similarity, doing so to gauge the sentence’s structure. However, it is reliant on semantic role fillers, parsers which indicate the grammatical function of a word within a sentence – be it an argument, predicate, modifier or any other feature. This creates the labeling that allows the metric to check whether the candidate translation matches the reference translation semantically and structurally. These are resources only available to some languages, a limitation that could make it less useful in a wider context. There exists a NOSRL version, which has fairly strong results, but may suffer from performance issues as there are many steps within the calculation [39]. However, this is outperformed by the ideal version WMD_O when considering the average correlation of all the languages, shown in Table 4.2.

	average correlation of all 7 languages
BLEND	0.633
MEANT_2.0	0.608
WMD_O , $\delta = \text{IDEAL}$, $\alpha = \text{IDEAL}$	0.599
MEANT_2.0-NOSRL	0.598
WMD_O , $\delta = 0.18$, $\alpha = 0.10$	0.594
CHRF++	0.579
AUTO DA	0.577
CHRF	0.577
WMD	0.576
BEER	0.571
TREEAGGREG	0.537
UHH_TSKM	0.486
BLEU2VEC_SEP	0.484
NGRAM2VEC	0.483
SENTBLEU	0.481

Table 4.2: Average correlation of different metrics in the WMT17 shared task against the proposed metrics, ranked in descending order. Trained/ensemble metrics are highlighted in grey.

4.2 Analysis

The WMD and WMD_O metrics provide a fairly strong basis for correlation with human scores. Focusing on semantic similarity works better than traditional purely string matching metrics, but still has some difficulties against metrics with more comprehensive training and modelling.

The results of WMD_O with $\delta = \text{IDEAL}$ are shown graphically in Figure 4.1, showing the correlations of the distribution. It should also be noted that the results of the IDEAL parameters do not vary too much in comparison with the results of the static parameter values, which are displayed in Figure 4.2.



Figure 4.1: WMD with word order penalty against Human scores for the seven language pairs. Parameters δ and α are the best values for each language pair.



Figure 4.2: WMD with word order penalty against Human scores for the seven language pairs. Parameters δ and α are set to static values of 0.18 and 0.10 respectively.

This suggests that the static parameters are a good basis for interpreting translation evaluation for any new language pair. It also suggests that the performance of the metric can be somewhat limited by its calculations as a finely tuned parameter will not provide a very significant performance enhancement over a less precise parameter choice.

Nevertheless, it can be seen that a pattern of clustering around the centre of each language pair’s graph is fairly evident. All of these are on a downwards trend line, showing the association between the WMD_O metric score and the given human score. Having such a dense group of points in the scatter graph also suggest that the metric is fairly capable of handling straightforward translations where the quality of the candidate translation is not overly contentious. Where the graphs have more points straying from the general downwards pattern is around the extreme ends of the human score, along the horizontal axis. These are sentence pairs where the scores of the annotators mark the translated sentence as either extremely good or extremely poor. Naturally, this creates more disputable results as opinions of a good or bad sentence can be affected by many factors, as mentioned previously. However, as the annotations of this dataset are the combination of 15 different assessments, the human scores of those points at extreme ends are likely to be reliable.

The range of the WMD metric is slightly variant for each language, as a result of the given dataset and the inherent properties of each language pair. The basic WMD is constrained between 0 and 1; with the additional word order penalty this increases to 0 and $1 + \delta$. Naturally those languages with higher values of δ in their IDEAL parameter, such as Turkish, will feature greater values of the WMD_O score. Allowing this value of δ to grow for Turkish-English prompts better correlation as points in more moderate values of WMD can be increased by a greater amount, lowering the discrepancy between sentences with very poor adequacy scores and those with poor fluency.

The effect of the fragmentation penalty is fairly evident when assessing how the penalty ratio of chunks to matched unigrams correlates with the human score; providing an indication of how fluency is important to a translation. Interestingly, the correlation is much better for some languages than others. This is slightly related to how much the fragmentation penalty impacted the correlation from WMD to WMD_O . Table 4.3 gives an indication of how the penalty ratios of each sentence correlated with the human score, while Table 4.4 shows the improvement between WMD and WMD_O for each language.

	cs-en	de-en	fi-en	lv-en	ru-en	tr-en	zh-en
Penalty ratio	0.419	0.406	0.520	0.411	0.457	0.574	0.522

Table 4.3: Correlation between penalty ratio and human score of each sentence for all seven language pairs.

	cs-en	de-en	fi-en	lv-en	ru-en	tr-en	zh-en
WMD	0.521	0.536	0.704	0.530	0.571	0.566	0.607
WMD_O	0.532	0.546	0.712	0.543	0.592	0.641	0.625
Difference	0.011	0.010	0.008	0.013	0.021	0.075	0.018
Percentage change (%)	2.11	1.87	1.14	2.45	3.68	13.3	2.97

Table 4.4: Improvement in performance of metric after implementing fragmentation penalty.

It can be seen that the improvement in performance of the Turkish language pair is much better than the other languages, although the others also have improvements in performance. Nevertheless, the addition of the fragmentation penalty does not do much to alleviate the impact of extreme values on the correlation of the majority of points within the central cluster. Figure 4.3 shows the results of WMD and WMD_O on the same graph. It can be seen that the overall shape of the points on the scatter graph in both colours remains very similar.



Figure 4.3: WMD (orange) and WMD_O (blue) results against the Human scores for the seven language pairs.

One of the reasons the results do not significantly change in the different iterations of the metric could be down to the dataset itself. The list of translations for each language all feature a fairly large number of sentences which contain out of vocabulary words, which could lead to words not having fully comprehensible representations with the zero vector resolution strategy. To analyse whether this is an impacting factor in the constant shape of the WMD metric results, sentences with out of vocabulary words are ignored in results of Table 4.5.

	cs-en	de-en	fi-en	lv-en	ru-en	tr-en	zh-en
Sentences considered (%)	70.4	68.4	48.6	50.4	64.6	52.6	61.3
WMD	0.525	0.537	0.709	0.536	0.575	0.569	0.609
WMD _O	0.500	0.556	0.641	0.526	0.535	0.513	0.594

Table 4.5: Results of metrics, ignoring out of vocabulary sentences.

The results of this indicate that having words which are out of the embedding’s vocabulary are not a major factor in the metric’s results. It should be noted that the results of WMD are better than WMD_O when ignoring these out of vocabulary sentences. This is potentially because the fragmentation penalty is best at handling sentences which are poorly translated, which naturally have more out of vocabulary words and a large number of chunks.

It is prudent then to analyse what the anomalous sentences skewing the correlation are. Graphically, these are the points which stray away from the trend line and central cluster. Close inspection of these specific sentences can provide an indication of where the metric falls down. When contrasted with the performance of a different metric, it can be seen that some of these anomalies are not present in the results of other metrics, suggesting an issue with the WMD metric rather than just the dataset. This direct comparison is in Figure 4.4.

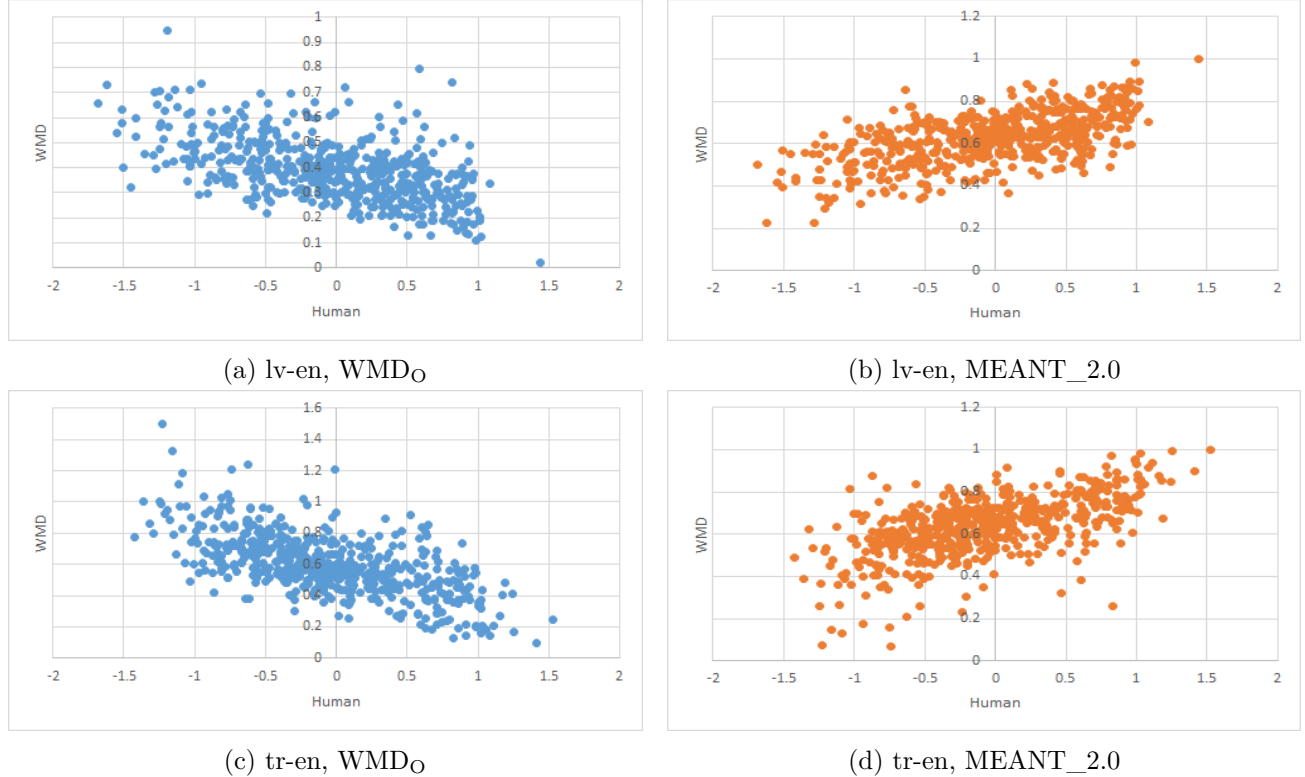


Figure 4.4: lv-en has better correlation for MEANT_2.0 and does not contain sparse points that WMD_O does; vice versa is true for tr-en. This shows how anomalous points weaken performance.

4.3 Anomalous sentences

Taking two sentence examples far from the trend line from each of the different languages, a few interesting results can be noted.

(1) Reference: The local wildlife will also not give them a break.

Candidate: And the wilderness will not be there.

Human score: -1.805

WMD_O score: 0.425

(2) Reference: That is useless.

Candidate: That's useless.

Human score: 1.154

WMD_O score: 0.423

Listing 4.1: Czech-English sentence pairs.

(3) Reference: The Association "sets an example for the entirety of Bavaria".

Candidate: The association is considered to be "Bavaria's most exemplary".

Human score: -0.294

WMD_O score: 0.586

(4) Reference: Mr Putin lashed out, accusing Ankara of stabbing Moscow in the
 ↪ back.

Candidate: Putin distributed and accused Ankara of, Russia in the backs pleases
 ↪ to be.

Human score: -1.787

WMD_O score: 0.375

Listing 4.2: German-English sentence pairs.

The most obvious discrepancy in some sentence pairs within this extract are the large amounts of untranslated words, which greatly impacts the final WMD metric. These are treated as out of vocabulary words and given the generic zero vector by the metric, and are punished accordingly with the missing word penalty. However, where these sentences struggle compared to other sentences with missing words is the sentence length. These are largely sentences which are short in length, often just a few words. Examples (5), (6), (7), and (10) are prime examples of this. The sentences are heavily punished by the missing word penalty because of their short length, which makes these points shoot upwards against the trend line. However, as these are generally bad translations, it may actually be that the human scores are too lenient on these translations. The lack of correlation does not necessarily mean a lack of agreement in translation evaluation, but a lack of nuance in defining differing degrees of translation quality.

This difference in sentence length is also a factor in overly punishing sentences without missing vocabulary, such as (2) and (9). The reference sentence is typically the longer sentence, means the candidate translation is likely not to have enough depth and substance in their semantics to convey the same message in such brevity. However, as the WMD metric needs to map every word from the reference to another in the candidate, some meanings are forcibly distributed to other words in the transportation problem, even if they do not make a large amount of sense. This makes the WMD value lower than it probably should be, when it should be higher to reflect the poor semantics. This is

(5) Reference: The education cuts continue.

Candidate: Koulutusleikkaukset continue.

Human score: -0.742

WMD_O score: 0.831

(6) Reference: The offence against Tuuli was a clear one.

Candidate: Tuula violate palpably.

Human score: -1.141

WMD_O score: 0.808

Listing 4.3: Finnish-English sentence pairs.

(7) Reference: Heat oil in a frying-pan.

Candidate: Pannā uzkaršē oil.

Human score: -1.197

WMD_O score: 0.947

(8) Reference: It runs from August 19 until October 30.

Candidate: The exhibition is seen from 19.August to 30.October.

Human score: 0.582

WMD_O score: 0.793

Listing 4.4: Latvian-English sentence pairs.

because some words in the reference, instead of mapping wholly to one word in the candidate sentence, might have to map a certain fraction to another word to make up the numbers in the constraints of the transportation problem.

Proper nouns also remain a problem, as mentioned previously. Examples (3), (4), (12), (13) and (14) all have names of people or places which are not part of the vocabulary. This makes determining an appropriate vector for these words very difficult.

Lastly, a key problem with the WMD metric in comparison to human scores is its ability to recognise appropriate sentence structures and general linguistic features in grammatically correct sentences. Example (4) has a candidate translation which is very disjointed and grammatically nonsensical. While there is a fragmentation penalty to handle the general fluency of the sentence, it can be seen that a lot of the chunks are quite consistent in both sentences; “*Mr Putin lashed out*” against “*Putin distributed*”, “*accusing Ankara of*” against “*accused Ankara of*”, “*Moscow in the back*” against “*Russia in the backs*”. The lack of fluency in this case is in terms of its grammar, which the implemented metric struggles to detect. This is compounded by the fact that many of the words used are semantically similar, matching the words across a very small distance in the semantic space. Another interesting feature of this sentence is their use of the comma as punctuation. As the metric preprocesses away this sort of punctuation, it is ignored in the calculation. However, in this example the comma is directly contributing to the lack of fluency and intelligibility of the sentence, which may impact the human score’s judgment.

This sentence pair is also problematic in its ending, which misses out the semantics of “*stabbing*”, replacing it with “*pleased*”; the candidate translation also inserts this section rather nonsensically into the sentence. This could potentially be a problem with the source language itself and the way it gets translated into the target language. Some language pairs are closer in terms of sentence structures and use of words, whereas others have vastly different structures. While the reference and candidate

(9) Reference: And there are a host of examples like this throughout the
 \hookrightarrow country.

Candidate: And such examples of the country mass.

Human score: -1.741

WMD_O score: 0.412

(10) Reference: Mr. Coe grimaced.

Candidate: Kou winced.

Human score: -0.793

WMD_O score: 0.609

Listing 4.5: Russian-English sentence pairs.

(11) Reference: "THOSE WHO COMMITTED THE COUP ATTEMPT SHOULD BE CLEARED FROM THE
 \hookrightarrow STATE"

Candidate: "Blow" GİRİŞİMİNDE BULUNANLARIN DEVLETTEN TEMİZLENMESİ GEREKİYOR

Human score: -1.227

WMD_O score: 1.504

(12) Reference: FETO ringleader Gülen's assets were seized

Candidate: Feö Elbaşı Gulen confiscated assets

Human score: -0.012

WMD_O score: 1.205

Listing 4.6: Turkish-English sentence pairs.

translations are clearly of a state-of-the-art level, the mismatch between the two languages could potentially create a lot of inconsistency in finding a “gold standard” translation to work off of. Two very similar languages are much more likely to end up having similar structures upon translation for both candidate and reference, both carrying the same semantics and message, while two more polarising languages could carry the same semantics and message but in two very contrasting styles, making it less adaptable to the WMD fluency mechanism. Even with the different parameters for each language pair, this sort of dissimilarity can still affect results.

Noticeably, a large proportion of these anomalous results are towards the negative side of the human score; that is to say translations which human annotators have judged to be poor candidates. The corresponding WMD_O scores for these are a mixture of high and low values, suggesting that there is some inconsistency in the metric’s judgment of poor translations. On the other hand, the metric is fairly good at matching the human scores for excellent translations, suggesting that it is easier to quantify the degree of how good a translation is as opposed to how bad a translation is.

	cs-en	de-en	fi-en	lv-en	ru-en	tr-en	zh-en
WMD _O	0.532	0.546	0.712	0.543	0.592	0.641	0.625
WMD _O (ignoring selected sentences)	0.537	0.548	0.715	0.551	0.589	0.645	0.632

Table 4.6: Results of WMD_O with and without ignoring selected sentence pairs.

Without these few selected sentences, the metric naturally performs better, as seen in Table 4.6. It is impertinent to selectively remove cases which weaken the results, but this does show that the effect of a couple of sentences can be very large on the overall results. It also gives an indication of where

(13) Reference: Helen Glover and Heather Stanning haven't lost a race in five
↪ years.

Candidate: Helen Graf and Heather Xi Solyariya in five years the outstanding
↪ defeats.

Human score: -1.727

WMD_O score: 0.443

(14) Reference: Trump spokeswoman Hope Hicks didn't immediately return a message
↪ seeking comment.

Candidate: The Trump spokesmen had not responded to this news.

Human score: 0.315

WMD_O score: 0.626

Listing 4.7: Chinese-English sentence pairs.

the metric is lacking in order to better improve results, and perhaps where this metric is limited in its construct or implementation.

Chapter 5

Evaluation

This chapter aims to give a holistic overview and evaluation of the project’s achievement. To clearly gauge the work of the implementation, the work done described against the objective of the project laid out in the introduction. This was to create an evaluation metric focusing on semantic similarity rather than string similarity, so that assessing the quality of translation can better preserve the message and tone of the original text. This would enable larger scale translation evaluation as two texts could be directly compared using neural word distributional models as a basis of a semantic space.

5.1 Metric performance

As described in the previous section, the quantitative performance of the metric is very competitive with the state-of-the-art metrics. It greatly surpasses the performance of simple string matching metrics and has strong results for the seven language pairs it was tested on, coming out top for two of the seven. This is a sign that the concept of WMD as a translation evaluation metric is relevant and even applicable to further language pairs.

While the basic WMD metric is straightforward to calculate and use, the improved WMD_O metric introduces the inconvenience of having two tunable parameters in the calculation, α and δ . These two parameters can be set per language for best results or kept at a static value in a trade-off for slightly worse correlations. The former would have to necessitate a large enough dataset to test the values on, which can be expensive as it requires human annotated data to compare correlations.

As elaborated in the previous chapter, the metric also has some inherent weaknesses in its construct. As it relies on a given neural word distributional model to calculate a WMD score, results are dictated by the quality of the embeddings used as well as the size of the vocabulary. Embeddings with lower dimension naturally have a lower quality as not enough semantic detail can be conveyed through each of the dimensions, while those with higher dimension generally reach a plateau after a given number, with extra dimensions only causing greater performance complexity [66]. Even embeddings with the same dimensionality can vary in performance as a result of its training parameters: the training data used, the type of model trained, the lower frequency limit are all factors that can make one model’s word vectors more meaningful than another’s.

Similarly, the impact that vocabulary size can have on performance of this WMD metric is significant. Embeddings with very few words, perhaps in the hundreds or thousands, are not likely to be able to account for every word in a translation sentence pair. Instead, many words are going to be left as out of vocabulary words and dealt with in a specific manner. Even using the 1 million vocabulary size fastText embedding, at least 30% of every language pair’s 560 sentence pairings contained a missing word.

Handling of out of vocabulary words is also something the metric does not necessarily do very well. The proposed strategy is to give every occurrence of an out of vocabulary word the zero vector, acting

as a constant neutral out of vocabulary value. However, this essentially treats all of these missing words in the same way; be it a proper noun or an untranslated word still in a foreign language. Ideally there would be no out of vocabulary vectors in the dictionary, but in the only method that enabled this, the fastText embeddings using n-grams, performance was poor. As the out of vocabulary rate was not necessarily high given the 1 million vocabulary size, setting this zero vector universally did not severely weaken results. Nevertheless, the best possible embeddings should be used for the best results.

Another weakness of the metric is its ability to deal with sentence pairs where the two sentences are of varying length. With the implemented fragmentation and missing word penalty, this is compounded when the sentence is very short, as each word is punished with extra severity. In addition, the makeup of the WMD calculation itself means that every word in the reference has to map to something in the candidate sentence; if the sentence lengths of both do not match then there are cases where the mappings are not bijective, causing some words to map to others just to fulfill the constraints despite being slightly nonsensical. To handle this the metric could be adapted to include penalties that take into account sentence length. The brevity penalty of BLEU or NIST could be a starting point for this.

Overall, the WMD metric has several aspects where it is weak, which can be improved upon and better tuned. However, WMD provides a strong basis for this change, as its results are already fairly promising compared to the state-of-the-art. Nevertheless, the metric is heavily reliant on the embedding it is provided, and this coupling is inevitable given the method of calculating semantic similarity. Should another method less reliant on linguistic resources exist to find similarity of words within a semantic space exist, the metric can still be used, but this appears unlikely. As the WMD_O version of the metric is merely a heuristic to the basic WMD, the general performance and shape of its results do not differ too greatly, but the scores do become more optimal.

5.2 Testing

Owing to the expensive nature of human annotation, the experimental dataset was largely focused on the seven language pairs into English of the WMT17 metrics task. This was limited to 560 sentence pairs per language, giving a total of 3920 sentences in total. While the French and German data initially tested had more sentences to reference, their human annotated scores were not well distributed, as many sentences were given the same score. For more variety in testing and a better overview of the situation, it was deemed more appropriate to just focus on the English data. Using exclusively English data also made it easier to understand calculations made when reading logs of the executed code, as this was a personally intelligible language. However, this certainly did limit the amount of data available to analyse the metric's success.

Lack of data also hampered the possibility of using the metric to attempt one of the other objectives of the project, directly comparing source texts to candidate translations. There did not exist an adequately large or sufficient dataset which gives a human annotated score to a candidate translation directly in comparison with the source sentence, rather than one which compares the candidate translation with a reference gold standard translation. This, coupled with time constraints, meant that it was not possible to use the metric to make this direct comparison. Nevertheless, the idea remains possible, which will be described in the following chapter.

As the results provided by WMT only show the raw score of each of the other metrics, without any information about the calculation process, it becomes slightly difficult to interpret the results of the other metrics and how they are able to perform better than the WMD metric. The correlations can be compared, as can the individual scores, to see how the metric ranks amongst others and whether other metrics face the same problems as WMD. Given the general pattern is similar for most of the language pairs, it suggests the dataset has some translations which are quite problematic for most metrics and will naturally weaken correlations. However, there are also points which drift away from the general trend line in WMD that do not in other metrics, showing their ability to handle these cases better

– but as the implementation details of these are not as readily available it is hard to tell how this happens.

5.3 Implementation

In terms of implementing the metric, there were several areas which could have been improved. In the initial stages of the project, the workflow consisted of making changes to the experimental settings, running the experiments over the entire dataset, then collating the results to calculate correlation and visualise the data graphically. These experimental settings were mostly things like the distance function and the amount of sentence preprocessing, as well as the out of vocabulary resolution strategies. This procedure was fairly time consuming, as each iteration of the experiment would take around half an hour to complete.

The time taken to execute one experiment iteration was down to two places in the code; loading the embedding and the actual WMD calculation itself. Given the size of each embedding is several gigabytes, loading this into memory usually took several minutes. Memory mapping techniques were used to speed this up so that each embedding only needed to be loaded once; though this was only applicable for embeddings which did not use the fastText n-gram approach. The actual calculation of WMD is of reasonable speed, and is of complexity $O(p^3 \log p)$, where p is the number of unique words in the sentence pairing. As this value is fairly low, the cubic scale does not hold back the calculation too much. However, this coupled with the use of embeddings are fairly CPU-dependent, which causes strain on computing resources. In addition, GPU optimisation is not available for training and using word2vec and fastText embeddings, making this resource a bottleneck. The amount of memory on the computer also had an impact on being able to load the embedding. As a result, the experiments had to be run on college computers, an extra inconvenience when working. Use of Git was key to ensure version control across multiple computers.

When tuning the parameters δ and α for WMD_O , the experiments did not need to be run multiple times, as the WMD_O result is only based on adjusting the chunk to matched unigram ratio by these weights. Collecting the ratio once was enough to adjust this repeatedly afterwards.

By using the PyEMD library, the calculation of WMD was limited to the library’s implementation, without any possible alteration or optimisation. This fast wrapper for Python includes some optimisations like thresholding the ground distance, but misses out on some optimisations described in the WMD paper [32] of word centroid distance or prefetching and pruning data.

One key to the smoothness of the implementation and analysis was the creation of logs to record the process of each WMD calculation. This helped greatly in making sense of the WMD calculation, removing it from the black box it came in and understanding what each parameter and variable was doing. Recording all this information was particularly helpful in analysis of anomalous results in the previous chapter, as the distance matrix and flow matrix of each sentence pair could be reviewed.

As a whole, the implementation of the project was smooth and fairly organised, but the actual execution of experiments unfortunately suffered from some bottlenecks. Despite this, the WMD experiments run were able to be done without much technical difficulty, and produce results which are both valid and competitive.

5.4 Usage

As the use of the metric was self-contained within the experiments run, there was never a need for this project to have a user-facing platform for one to test the metric on. This is potentially something which would have made gathering qualitative feedback easier from users, as a clean frontend for this metric would create a good platform for people to query the WMD score for any arbitrary sentence

pair. Creating a program or API to help visualise this may have helped the presentation of the metric, although this does remain secondary to the metric's actual functionality.

5.5 Qualitative feedback

The contents of the project have also gone through evaluation from established reviewers in the field. This metric has been submitted to the WMT19 conference [10] along with a brief paper summarising the gist of the metric's work and results. As part of the process, the Program Committee reviewed the work on a 1 to 5 scale on the following criteria: Relevance, Soundness / Correctness, Clarity, Meaningful Comparison, and Overall Recommendation. The feedback of the two reviewers, along with their detailed comments, are included in Listing 5.1.

```
=====
                        REVIEWER #1
=====
The authors present an extension to the Word Mover's Distance metric for MT
↪ evaluation to take word order into account to include fluency in the score.
↪ They accomplish this by including a weighted fragmentation penalty into the
↪ score, similar to the one in the METEOR metric. The results show improved
↪ correlation with human judgments when compared to the original WMD score. This
↪ gain comes however at the expense of having a tunable parameter in the
↪ calculation of the score, which is likely to be language-specific.
                        Relevance (1-5): 5
                Soundness / Correctness (1-5): 4
                        Clarity (1-5): 3
                Meaningful Comparison (1-5): 4
                Overall Recommendation (1-5): 4

=====
                        REVIEWER #2
=====
The paper applies a evaluation method (WMD) that has been used to measure document
↪ distance to MT translation evaluation, and modifies the method by introducing a
↪ heuristic penalty to address word order. Experiments show that it is better
↪ than many evaluation metrics using surface string matching but still seems to
↪ fall behind of, e.g., MEANT 2.0, that also uses fuzzy embedding matching. The
↪ method that uses a transformation to measure distance for eval MT seems
↪ interesting to me and may serve as a framework for further improvement.
                        Relevance (1-5): 5
                Soundness / Correctness (1-5): 3
                        Clarity (1-5): 4
                Meaningful Comparison (1-5): 4
                Overall Recommendation (1-5): 4
```

Listing 5.1: Reviews of WMD metric from the Program Committee of the WMT19 conference.

From this it can be seen that the overall view of this metric is positive, with strong agreement on its relevance to translation evaluation. In particular, the second reviewer mentions the promise of using transformation to measure distance in evaluating machine translations. This is essentially the

Human	WMD	Review 1	Review 2	Review 3	Review 4	Review 5
Good	Good	Good	Good	Good	Good	Good
Good	Good	Good	Good	Moderate	Good	Good
Good	Moderate	Good	Good	Good	Good	Moderate
Good	Good	Moderate	Good	Good	Good	Good
Good	Good	Good	Good	Good	Good	Good
Moderate	Good	Moderate	Moderate	Moderate	Moderate	Moderate
Moderate	Moderate	Moderate	Good	Moderate	Moderate	Moderate
Moderate	Moderate	Good	Moderate	Moderate	Moderate	Moderate
Moderate	Bad	Bad	Moderate	Bad	Moderate	Moderate
Moderate	Moderate	Moderate	Moderate	Moderate	Moderate	Moderate
Bad	Bad	Moderate	Bad	Bad	Bad	Bad
Bad	Bad	Bad	Bad	Bad	Bad	Bad
Bad	Bad	Bad	Bad	Bad	Bad	Moderate
Bad	Moderate	Moderate	Moderate	Bad	Bad	Bad
Bad	Bad	Bad	Moderate	Bad	Bad	Bad

Table 5.1: Survey results of five peers asked to review 15 sentence pairs from the Chinese-English dataset.

use of the metric to compare a source text to a candidate translation, which will be elaborated upon in the conclusion.

Outside of this, feedback was also sought from peers to review the overall sanity of the metric and approach. To make this process simple and straightforward, the questions asked were simple categorisation problems, focused on whether the metric worked well on a broader scale rather than in slight nuances. This broader scale split a sentence pair into three categories: a good translation, a moderate translation, or a bad translation. Sentence pairs were selected where the human score and WMD score were in agreement in this categorisation, as well as pairs where they were not. To prevent ambiguity, only translations well into the upper or lower third of scores were selected as good or bad translations respectively. There were 15 sentences selected in total. The demographics of this qualitative survey were a group of five bilingual students, fluent in both English and Chinese. As a result Chinese-English dataset was the focus of survey, to prevent any misunderstandings due to mistranslated words or lack of context. The results of this survey are shown in Table 5.1.

The results of this indicate that there is a general consensus around the labelling of sentence pairs. Most pairs in which the human score and WMD score match are also matched by the reviewers, while those with a slight variation also see a slight variation from the reviewers. Most noticeably, there is no great divide over the categorisation of sentences; no sentence pair which skips over the moderate categorisation and polarises between good and bad. This is a good sign for the metric’s growth as it suggests that scores are not volatile. It also shows that the correlation between these scores and the human score is well founded and agreed upon by random users.

Chapter 6

Conclusions & Future Work

The work in this project has achieved the goal of proposing a novel method of evaluating machine translations, focusing on word embeddings and the semantic space. To make the task achievable in the given timeframe, its scope had to be reduced to the creation and tuning of this metric, rather than any application of it to direct translation evaluation problems between a source text in one language and a candidate translation in another, as outlined in the objective. The metric created has been able to make use of large scale word embeddings of high dimensionality to measure semantic similarity. It has also been able to utilise the innovative Word Mover’s Distance to compare values within the semantic space. This allows sentences to be compared by methods other than traditional string matching metrics, providing far more flexibility and semantic understanding to the translation evaluation process.

The results of this work are encouraging, with the metric performing strongly against state-of-the-art metrics. In particular, the metric greatly outperforms known string matching metrics, only losing out to the trained ensemble metric BLEND and the linguistic resource dependent MEANT_2.0. Of course, there remain known weaknesses in the metric; short sentences, embedding quality, out of vocabulary words to name a few, but both the WMD and WMD_O iterations of the metric provide strong results that are able to perform closely with gold standard human evaluations. The improved WMD_O version of the metric is also able to handle the issue of word order that is neglected by the baseline WMD’s bag-of-words approach, but it does necessitate the tuning of two parameters α and δ – though static values for these have been proven to provide similar results.

The hope is that the metric designed and implemented in this work provides insights to how the semantic space can be used within the machine translation evaluation field, as well as how it can then be extended to other natural language or translation tasks. It was found that combining insights from previous literature along with creation and experimentation of different components enabled effective solution of the initial challenge to create an automatic evaluation metric that focused on word meaning while still maintaining a semblance of fluency.

6.1 Lessons learnt

A key to take away from implementing this type of project is the importance of organisation. This applies for the writing of code, collation of results, or even the writing of the report. However, where this most applies is the handling of data and resources. With the many different experimental settings run during this type of work, it was important not to mix up any results and make sure each setting was being run with what it was labelled as and not anything else. This ensured integrity of results, paramount to development of the metric. This was initially a big problem before a good structure was set up for the project; as the project grew a configuration file was set up to alter these variables in the code programmatically rather than manually to avoid anything being neglected.

The use of WMD as a distance measure within the semantic space was a good starting point for development of an evaluation metric, which set up the possibilities of word order penalties and fragmentation penalties. The word order penalty first appeared more intuitive, but it became clear that it was not a good method to evaluate fluency as compared to the fragmentation penalty. This was a penalty inspired by previous literature, showing the importance of building and adapting successful approaches. While the idea of the word order penalty did not work here, the work could remain useful somewhere along the line for considering word order in a different context.

Another critical take from the project is the need to handle out of vocabulary words in a semantic embedding focused metric. Simple string based methods do not have this issue, as there is no dictionary to speak of. Semantic word vectors are the building blocks of this metric and need to be counted for each occurring word in the sentence pair. There are a multitude of strategies to handle this; the one chosen for its best performance in this case was the zero vector. There may have been other single vectors that would have produced better results but this was the most straightforward choice and required no tuning. The approach of fastText n-grams appeared promising as it meant there would be no out of vocabulary words; however this was not able to combine well with WMD and produced poor results. This may have been due to issues with the many mistranslated words in the dataset's sentence pairs, which would not have made sense to use n-grams to piece back together. This n-gram approach remains an available avenue for future work in the area.

This issue of the embedding used is also critical to the success of WMD; the better quality the embedding is the better results will be. The results in this set of experiments found the best performing embedding to be the pre-trained fastText embedding of vocabulary size 1 million. Different datasets may be more optimised for different embeddings or even their own trained embeddings. While training embeddings was pursued in this project it was not a heavily considered option due to the computing sacrifice and cost needed to train just one embedding on a corpus just a fraction of the size that pre-trained embeddings train on. The embeddings which were trained in this experiment may have been done using suboptimal parameters, leading to poorer performance.

It is also clear from the implementation of this metric that certain values of α and δ work better on different language pairs, which begs the question of how a new language pair will adapt to the metric, or even how the same language pair will adapt with a new set of sentence pairs as data. This could be tuned for each language pair with a test set of data, or the static values could be used at a slight performance loss – which would still likely outperform the baseline WMD.

6.2 Future Work

To improve the metric in the future key weaknesses can be tackled. The most solvable of these is the sentence length discrepancy which creates the anomalies seen in given examples. This could make use of a brevity penalty or any other factor that takes into account difference in sentence length. If implemented, this should improve results for all languages by bringing those points further away from the general pattern closer to the trend line. Other factors such as embedding quality are less feasible to change, although they will also bring improved results.

This work within semantic spaces can also be extended to the second part of the objective; allowing translation evaluations to be directly carried out between a source text in one language to a candidate translation in a different language. Comparisons of two segments are currently done within the monolingual vector space. In the case of these experiments, this is the English vector space. Future translation evaluations can make use of what is known as cross-lingual embedding spaces to carry out the same calculations. This carries vectors for two languages in a single vector space, so one space would represent two languages, and carry words in the vocabulary of both languages. This should place similar words in both languages in the same area of the vector space, so that comparison can be made as if all the words were part of one common monolingual space as done throughout this project. Figure 6.1 describes how the embeddings of two languages are transformed into one cross-lingual space. Work

into cross-lingual embeddings has been growing in recent years [14] and this metric could leverage the potential of this area in the future to further improve automatic translation evaluation.

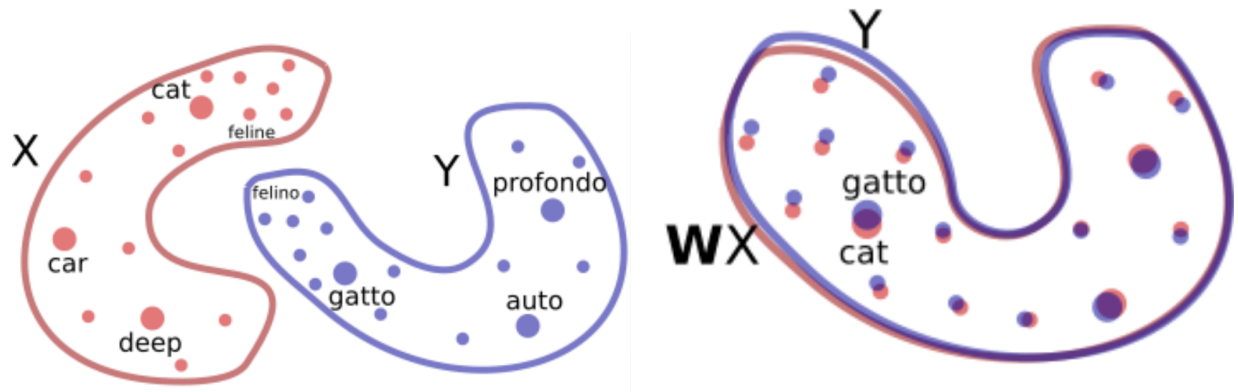


Figure 6.1: Transformation of embeddings of two languages X and Y (left) into one cross-lingual embedding W (right).

Bibliography

- [1] Satanjeev Banerjee and Alon Lavie. “METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgments”. In: *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization* (2005), pp. 65–72.
- [2] Marco Baroni, Georgiana Dinu, and German Kruszewski. “Don’t count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors”. In: *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics* 1 (2014), pp. 238–247.
- [3] Yoshua Bengio. “Neural net language models”. In: *Scholarpedia* 3.1 (2008), p. 3881.
- [4] Yoshua Bengio et al. “A Neural Probabilistic Language Model”. In: *Journal of Machine Learning Research* 3 (2003), pp. 1137–1155.
- [5] Christopher Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.
- [6] Ondrej Bojar, Yvette Graham, and Amir Kamran. *Results of the WMT17 Metrics Shared Task*. 2017. URL: <http://www.statmt.org/wmt17/pdf/WMT55.pdf> (visited on 05/05/2019).
- [7] Ondřej Bojar, Yvette Graham, and Amir Kamran. *EMNLP 2017 Second conference on Machine Translation Shared Task: Metrics*. 2017. URL: <http://www.statmt.org/wmt17/metrics-task.html> (visited on 05/03/2019).
- [8] Ondřej Bojar, Yvette Graham, and Amir Kamran. *EMNLP 2017 Second conference on Machine Translation Shared Task: Multimodal*. 2017. URL: <http://www.statmt.org/wmt17/multimodal-task.html> (visited on 05/03/2019).
- [9] Ondrej Bojar et al. “Findings of the 2013 Workshop on Statistical Machine Translation”. In: *Proceedings of the Eighth Workshop on Statistical Machine Translation* (2013).
- [10] Ondřej Bojar et al. *Fourth Conference on Machine Translation*. 2019. URL: <http://www.statmt.org/wmt19/metrics-task.html> (visited on 06/16/2019).
- [11] Michael R Bussieck, Arne Stolbjerg Drud, and Alexander Meeraus. “A New Quantitative Quality Measure for Machine Translation Systems”. In: *In Proceedings of COLING-1992* (1992).
- [12] John B Carroll. “An Experiment in Evaluating the Quality of Translations”. In: *Mechanical Translation and Computational Linguistics* 9.3 (1966).
- [13] Stanley F Chen and Joshua Goodman. “An Empirical Study of Smoothing Techniques for Language Modeling”. In: (1998).
- [14] Alexis Conneau et al. “Word Translation Without Parallel Data”. In: *arXiv preprint arXiv:1710.04087* (2017).
- [15] Scott Deerwester et al. “Indexing by Latent Semantic Analysis”. In: *Journal of the American Society for Information Science* (1990).
- [16] Michael Denkowski and Alon Lavie. “Meteor 1.3: Automatic Metric for Reliable Optimization and Evaluation of Machine Translation Systems”. In: *Proceedings of the EMNLP 2011 Workshop on Statistical Machine Translation* (2011).

- [17] Michael Denkowski and Alon Lavie. “METEOR-NEXT and the METEOR Paraphrase Tables: Improved Evaluation Support For Five Target Languages”. In: *Proceedings of the ACL 2010 Joint Workshop on Statistical Machine Translation and Metrics MATR* (2010).
- [18] Michael Denkowski and Avon Lavie. “Choosing the Right Evaluation for Machine Translation: an Examination of Annotator and Automatic Metric Performance on Human Judgment Tasks”. In: *Proceedings of the 12th Conference of the Association of Machine Translation in the Americas* (2010).
- [19] George Doddington. “Automatic evaluation of machine translation quality using n-gram co-occurrence statistics”. In: *Proceedings of ARPA Workshop on Human Language Technology* (2002).
- [20] Desmond Elliott et al. *Findings of the Second Shared Task on Multimodal Machine Translation and Multilingual Image Description*. 2017. URL: <http://www.statmt.org/wmt17/pdf/WMT18.pdf> (visited on 05/05/2019).
- [21] Jean-Philippe Fauconnier. *French Word Embeddings*. 2015. URL: <http://fauconnier.github.io>.
- [22] John R Firth. “A synopsis of linguistic theory 1930-1955”. In: *Studies in Linguistic Analysis* (1957), pp. 1–32.
- [23] Jonas Gehring et al. “Convolutional Sequence to Sequence Learning”. In: *arXiv preprint arXiv:1705.03122v2* (2017).
- [24] Genevieve Gorrell. *Latent Semantic Analysis: How does it work, and what is it good for?* 2005. URL: http://www.dcs.shef.ac.uk/~genevieve/lisa_tutorial.htm (visited on 02/24/2019).
- [25] Yvette Graham, Nikita Mathur, and Timothy Baldwin. “Accurate Evaluation of Segment-level Machine Translation Metrics”. In: (2015).
- [26] Yvette Graham et al. “Measurement of Progress in Machine Translation”. In: *Proceedings of the Australasian Language Technology Workshop 2012* (2012).
- [27] Natural Language Processing for Hackers. *Complete Guide to Word Embeddings*. 2019. URL: <https://nlpforhackers.io/word-embeddings/> (visited on 06/17/2019).
- [28] Zellig S Harris. “Distributional Structure”. In: *Word* (1954), pp. 146–162.
- [29] Frederick Jelinek and Robert L Mercer. “Interpolated estimation of Markov source parameters from sparse data”. In: *Proceedings of the Workshop on Pattern Recognition in Practice* (1980).
- [30] Philipp Koehn and Christof Monz. “Manual and Automatic Evaluation of Machine Translation between European Languages”. In: *Proceedings of the Workshop on Statistical Machine Translation* (2006), pp. 102–121.
- [31] Paul Kroeger. *Analyzing Grammar*. Cambridge University Press, p. 248. ISBN: 978-0-521-81622-9.
- [32] Matt J Kusner et al. “From Word Embeddings To Document Distances”. In: *Proceedings of the 32nd International Conference on International Conference on Machine Learning 37* (2015), pp. 957–966.
- [33] Alon Lavie and Abhaya Agarwal. “Meteor: An Automatic Metric for MT Evaluation with High Levels of Correlation with Human Judgments”. In: *Proceedings of the ACL 2007 Workshop on Statistical Machine Translation* (2007).
- [34] Alon Lavie and Abhaya Agarwal. “METEOR, M-BLEU and M-TER: Evaluation Metrics for High-Correlation with Human Rankings of Machine Translation Output”. In: *Proceedings of the ACL 2008 Workshop on Statistical Machine Translation* (2008).
- [35] Alon Lavie and Michael Denkowski. “The METEOR Metric for Automatic Evaluation of Machine Translation”. In: *Machine Translation* (2010).

- [36] Omer Levy and Yoav Goldberg. “word2vec Explained: Deriving Mikolov et al.’s Negative-Sampling Word-Embedding Method”. In: *arXiv preprint arXiv:1402.3722*. (2014).
- [37] Omer Levy, Yoav Goldberg, and Ido Dagan. “Improving Distributional Similarity with Lessons Learned from Word Embeddings”. In: *Transactions of the Association for Computational Linguistics* 3 (2015), pp. 211–225.
- [38] Chin-Yew Lin and Franz J Och. “Automatic Evaluation of Machine Translation Quality Using Longest Common Subsequence and Skip-Bigram Statistics”. In: *Proceedings of ACL* (2004).
- [39] Chi-Kiu Lo. “MEANT 2.0: Accurate semantic MT evaluation for any output language”. In: *Proceedings of the Conference on Machine Translation (WMT)* 2 (2017), pp. 589–597.
- [40] Fourth Conference on Machine Translation (WMT19). *Shared Task: Machine Translation of News*. 2019. URL: <http://www.statmt.org/wmt19/translation-task.html> (visited on 06/17/2019).
- [41] William Mayner. *Fast EMD for Python: a wrapper for Pele and Werman’s C++ implementation of the Earth Mover’s Distance metric*. <https://github.com/wmayner/pyemd>. 2019.
- [42] Tomas Mikolov. “Language Modeling for Speech Recognition in Czech”. In: *Masters thesis, Brno University of Technology* (2007).
- [43] Tomas Mikolov et al. “Advances in Pre-Training Distributed Word Representations”. In: *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*. 2018.
- [44] Tomas Mikolov et al. “Distributed Representations of Words and Phrases and their Compositionality”. In: *Proceedings of the 26th International Conference on Neural Information Processing Systems* 2 (2013), pp. 3111–3119.
- [45] Tomas Mikolov et al. “Efficient Estimation of Word Representations in Vector Space”. In: *ICLR Workshop* (2013).
- [46] Tomas Mikolov et al. *Language Modeling with Recurrent Neural Networks*. 2011. URL: http://www.fit.vutbr.cz/research/groups/speech/servite/2012/mikolov_RNN_LM.pdf (visited on 02/24/2019).
- [47] Andriy Mnih and Geoffrey Hinton. “Three New Graphical Models for Statistical Language Modelling”. In: *International Conference on Machine Learning* (2007).
- [48] Kishore Papineni et al. “BLEU: a Method for Automatic Evaluation of Machine Translation”. In: *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics* (2002), pp. 311–318.
- [49] Ofir Pele and Michael Werman. “A linear time histogram metric for improved sift matching”. In: *Computer Vision–ECCV 2008*. Springer, 2008, pp. 495–508.
- [50] Ofir Pele and Michael Werman. “Fast and Robust Earth Mover’s Distances”. In: *IEEE International Conference on Computer Vision* (2009).
- [51] Ma Qingsong et al. “Blend: a Novel Combined MT Metric Based on Direct Assessment”. In: *Proceedings of the Conference on Machine Translation (WMT)* 2 (2017), pp. 598–603.
- [52] CJ van Rijksbergen. *Information Retrieval*. Butterworths, 1979.
- [53] Yossi Rubner, Carlo Tomasi, and Leonidas J Guibas. “A Metric for Distributions with Applications to Image Databases”. In: *IEEE International Conference on Computer Vision* (1998).
- [54] Yossi Rubner, Carlo Tomasi, and Leonidas J Guibas. “The Earth Mover’s Distance as a Metric for Image Retrieval”. In: *International Journal of Computer Vision* (2000), pp. 99–121.

- [55] Adriaan MJ Schakel and Benjamin J Wilson. “Measuring Word Significance using Distributed Representations of Words”. In: *CoRR* abs/1508.02297 (2015). URL: <http://arxiv.org/abs/1508.02297>.
- [56] Holger Schwenk and Jean-Luc Gauvain. “Training Neural Network Language Models On Very Large Corpora”. In: *Joint Conference HLT/EMNLP* (2005), pp. 201–208.
- [57] Grigori Sidorov et al. “Soft Similarity and Soft Cosine Measure: Similarity of Features in Vector Space Model”. In: *Computación y Sistemas* (2014), pp. 491–504.
- [58] Matthew Snover et al. “A Study of Translation Edit Rate with Targeted Human Annotation”. In: *Proceedings of the 7th Biennial Conference of the Association for Machine Translation in the Americas* (2006), pp. 223–231.
- [59] Christoph Tillmann et al. “Accelerated DP based search for statistical translation.” In: *Proceedings of EUROSPEECH* (1997).
- [60] Joseph P Turian, Luke Shen, and Dan I Melamed. “Evaluation of Machine Translation and its Evaluation”. In: *Proceedings of MT Summit IX* (2003).
- [61] Ashish Vaswani et al. “Attention Is All You Need”. In: *31st Conference on Neural Information Processing Systems* (2017).
- [62] David Vilar et al. “Human Evaluation of Machine Translation Through Binary System Comparisons”. In: *Proceedings of 2nd Workshop Statistical Machine Translation* (2007), pp. 96–103.
- [63] Ye-Yi Wang, Alex Acero, and Ciprian Chelba. “Is word error rate a good indicator for spoken language understanding accuracy”. In: *IEEE Workshop on Automatic Speech Recognition and Understanding* (2003).
- [64] John S White, Theresa O’Connell, and Francis O’Mara. “The ARPA MT Evaluation Methodologies: Evolution, Lessons, and Future Approaches”. In: *Proceedings of the first conference of the Association for Machine Translation in the Americas* (1994), pp. 193–205.
- [65] *word2vec*. 2013. URL: <https://code.google.com/archive/p/word2vec/> (visited on 02/24/2019).
- [66] Zi Yin and Yuanyuan Shen. “On the Dimensionality of Word Embedding”. In: *32nd Conference on Neural Information Processing Systems* (2018).
- [67] Francisco Zamora-Martinez, Maria J Castro-Bleda, and Salvador Espana-Boquera. “Fast evaluation of connectionist language models”. In: *International Conference on Artificial Neural Networks* (2009), pp. 144–151.