university of groningen / faculty of arts

Classification of Human vs Machine Translated Text

A binary classification task

Wessel Reijngoud

Bachelor thesis
Informatiekunde
Wessel Reijngoud
s2580748
June 21, 2018

ABSTRACT

Machine translation has, since its birth, improved a lot and has thus raised questions. Most of these questions are about how it compares to human translation and if it is possible to detect whether a text is translated by a machine or by a human. Multiple techniques and algorithms have been created to automate the process of classifying a text as being human translated or machine translated. This thesis will focus on the classification of machine translated or human translated text for Dutch. It will also focus on the most informative features from such classifier. The results were that such a classifier can be built for Dutch with an accuracy score of 0.78. The most informative features contained some interesting synonyms that were typical for either the human or the machine translation.

CONTENTS

PREFACE

The time that I never thought would come has come; My bachelor thesis! I enrolled myself for information science in 2013, but I had no real clue why. I always had my interest in computers and not knowing what to study, I enrolled myself for this. I didn't know what I was getting myself into and I barely knew what the study meant.

The first couple of years I was busier with the life of a student outside of studying. The first introductory courses about programming and databases weren't appealing to me at all and it was only in my third year that I found out that information science and acquiring knowledge about it was fun. I got interested by courses that actually made me produce something interesting. Too bad though that because of my lack of interest in the introductory classes that I lacked skills in programming. Now though I am writing my thesis which means I have gotten my skills on par again. I have learned that doing research on your own is challenging, but fun.

I would like to thank my supervisor A. Toral for putting up with me during these last months and helping me greatly where needed. I would also like to thank my friends who made studying here great fun, but also helping me out with issues I had.

1 INTRODUCTION

Machine translation (MT) has, since its birth, improved a lot and has thus raised questions. Most of these questions are about how it compares to human translations (HT) and if it is possible to detect whether a text is translated by a machine or by a human. Of course the old statistical machine translation (SMT) systems from the early 2000's created texts that are easily detected as MT by a human, because of their morphological and lexical errors (Bentivogli et al., 2016). Nowadays though, most systems produce good translations that are almost identical to something a human would produce because most of the systems now have a neural network approach. As noted in Bentivogli et al. (2016), neural machine translation (NMT) is better than SMT because, NMT copes better with lexical diversity than SMT. NMT also makes less morphological and lexical mistakes than SMT. Classifying whether a text has been translated by a human or by a machine can be used to detect fraudulent websites. These websites are almost identical to the real websites but they sell products that are fake or are fraudulent in other ways. Texts on these websites are usually in English but have not been originally produced in English. They have been translated from their source language to English by a machine translation system. Classifying texts on these websites as being machine translated can be a step forward in tagging such websites as fraudulent. This way the consumer can be notified whether the site is trustworthy or not.

This thesis will focus on the classification of MT or HT text for Dutch, taking into account the source language. It will also focus on the most informative features from such classifier. Specifically, these 2 research questions will be addressed:

1. Can a human-machine translation classifier be built for the Dutch language?
2. What are the most informative features of both classes?

The answers to these questions will be introduced by first looking at previous research on the topic of text classification. Secondly, the data and material and how they were acquired will be discussed in detail. After that the methodology of how the classifier was built and why certain choices have been made will be discussed. At last, the results will be presented and discussed and conclusions will be drawn from these results.

2 BACKGROUND

In this chapter, earlier research on the topic of text classification will be discussed to provide motivation for the research topic. Also some interesting features that others have used will be dicussed.

As noted in Ahrenberg (2017), MT technology has been improving a lot, thus the question arises if there is still a way to discriminate between HT and MT. Machines need examples of differences to be able to learn specific features of one or the other translation. Ahrenberg wanted to extract useful features that best discriminate the differences between Swedish human translated text and the Swedish machine translated text with an English text as its source language. Ahrenberg specifically looked at shift- or order differences in the translation processes. Ahrenberg showed that a MT is more similar to the source language in ways such as, length, structure and information flow. Another way of classifying HT or MT is done by Li et al. (2015), where they look at linguistic features without needing the source language to classify as HT or MT. Li et al. specifically used features that look at the syntactic structure of sentences. In contrast to Ahrenberg, Li et al. did not have to have the source language available to discriminate between the two translations. Li et al. also used an interesting feature where they looked at the density of function words and pronouns, where they found that SMT systems made mistakes.

Just like in Ahrenberg (2017) this thesis will look at features that need to have the source language available to look at differences in structure and length. His work however has focused on translation to Swedish in particular. This thesis will focus on the translation to Dutch and it will use the feature for difference in sentence length just like in Ahrenbrg. Just like in Li et al. (2015) some features used in this thesis will focus on structure of the sentences, more specifically at the difference in length, in the amount of dots and in the amount of commas between HT and MT. This thesis will focus purely on the translation of the Europarl corpus to Dutch by both human and machine. For classification as either HT or MT, the support vector algorithm (SVM) first introduced by Cortes and Vapnik (1995) will be used.

3 DATA AND MATERIAL

3.1 COLLECTION

This thesis will focus on classifying human and machine translated text in Dutch, because there is already an abundance of research done in most other languages.

The data set selected for this thesis is part of the parallel *Europarl* corpus for Dutch and English (Koehn, 2005). This corpus contains sentences from the European Parliament. The data set is parallel as in that the sentences in the Europarl corpus are aligned in the English and Dutch files. This corpus was chosen because of its availability in 21 European languages and the enormous amount of sentences that is included in the corpus. The data collected from the Europarl corpus make up the original English data and the human translated Dutch data in the data set.

The other part of the data set, the machine translated text, has been created by using the Googletrans API (Susik and Han, 2015). This API is an unofficial Google Translate API. The choice for using an unofficial API instead of the official Google Translate API is because the official one costs a lot of money and the unofficial one is free to use. The GoogleTrans API translates given input to any specific target language. Using the Googletrans API, 10059 sentences from the original English Europarl data have been translated to Dutch. Only 10059 sentences have been translated because the GoogleTrans API is quite slow. Translating only this part of the data set already took around 5 hours to do.

Table 1: Some example sentences from the data.

Source (English)	HT (Dutch)	MT (Dutch)
Resumption of the session	Hervatting van de zitting	Hervatting van de sessie
Please rise, then, for this minute' s silence.	Ik wil u vragen deze minuut stilte staande in acht te nemen.	Sta alsjeblieft op voor de stilte van deze minuut.
Madam President, on a point of order.	Mevrouw de Voorzitter, ik wil een motie van orde stellen.	Mevrouw de Voorzitter, een motie van orde.

Because the GoogleTrans API is an unofficial Google Translate API, it has some flaws. After inspecting the data it seemed that Googletrans had timed out a few times while translating the whole set and left gaps in the data. There were a total of 164 gaps in the data, so the final amount of sentences translated by the Googletrans API were 9895. Because the data set has to be parallel for the given task in this thesis, the English and human Dutch translated data have also been reduced to 9895 sentences. This realigned all the sentences in the data set.

Table 1 shows three sentences that occur in the data for each of the translations and the source. Table 2 shows the amount of words and unique words in the original English (OG) data, the HT and the MT data.

Table 2: The amount of words and unique words in the data.

Translation	Sentences	Words	Unique Words
OG	9895	248746	10826
HT	9895	242112	14451
MT	9895	248603	13857

3.2 ANNOTATION

Labels that can be assigned to the data in this particular thesis research are sentences either being HT or MT. OG is not a label, but it is used to create numerical features to identify differences between the human and machine translated text. Because the data set is parallel, there are 9895 HT sentences and labels and 9895 MT machine translated sentences and labels which sum up to 19790 sentences divided equally over 2 labels. The features that must indicate these labels are bag of words, difference in sentence length, difference in the amount of dots (Ndots) and the difference in the amount of commas (Ncommas). These will be further explained in 4.2.

3.3 PROCESSING

In this section, the formatting of the data and feature files will be explained.

3.3.1 Formatting Data

In order to get the data to be used by a classifier it needs to be in the correct data format. The OG, HT and MT sentences harvested were all in separate text files. These text files can be read individually by the classifier but as mentioned in 3.1 the GoogleTrans API left some gaps due to timeouts. So the text files have been put together into a comma separated values (.csv) file, containing three columns, the first column containing the OG sentences, the second column containing the HT sentences and the third column containing the MT sentences. Combining them into a .csv file made it easy to exclude the gaps the GoogleTrans API left because of timeouts mentioned in 3.1, by dropping all the rows which had empty cells in them. This realigned all the sentences in all of the rows. By putting all the data into a .csv, the corresponding sentences can be easily extracted per class from one file during classification.

3.3.2 Formatting Features

The numerical features, which will be further touched upon in 4.2, used during classification were also put together in one .csv file. The three different features extracted, all ended up in separate .csv files. Those were then also combined into one .csv file with each column corresponding to a feature containing numerical values where the first 9895 rows corresponded to the HT data and the rows 9896 up to 19790 corresponded to the MT data. This way all of these features can be used individually and in every combination possible to check their influence on the classifier scores.

4 METHODOLOGY

As mentioned before, the goal of this thesis is to classify texts as being translated by a human or translated by a machine. The data set will be split using 10-fold cross validation as explained in Refaeilzadeh et al. (2009), this divides the data set up in 10 parts and runs the classification 10 times. It uses 1/10 parts for testing and the rest for training, but every iteration it uses a different part for testing and different parts for training. This way all of the parts from the data set get used for testing once. The tested classifiers will be evaluated by means of F1-score and accuracy.

For this thesis the Python programming language will be used for all programs. For building and testing classifiers, the Scikit Learn module will be used as it is easy and efficient to use (Pedregosa et al., 2011). All data and source codes needed to reproduce results from this thesis are available on a GitHub repository (Reijngoud, 2018).

4.1 CLASSIFIER

4.1.1 Baseline

To be able to evaluate the results of the classifier, some point of reference is needed. For this purpose a baseline classifier will be built. The baseline classifier will randomly assign labels to the sentences in the test set. The accuracy that follows from this gives an indication of how much of the correct classification is caused by chance. A classifier from the Scikit-learn package that does this is the DummyClassifier and it has been used to produce baseline scores.

4.1.2 Approach

Text classification uses algorithms that learn from a pre-labeled data set called the training set and this way tries to learn differences between labels. Then it applies this to an unlabeled set of data, the test set, and tries to classify these into one of the labels. There are multiple popular techniques for classification for between two-group classification tasks. Naive bayes and SVM seem to be leading for use in classification between HT and MT (Kumar Yogi et al., 2015). For traditional topic classification with text an algorithm called SVM invented by Cortes and Vapnik (1995) seems to be the most popular. To use all data in predicting the right classes in the results, n-fold cross validation is used. Cross validation is very popular for use in estimating the performance in learning algorithms like a classification task (Refaeilzadeh et al., 2009).

4.1.3 Choosing a classifier

For all classification tasks in this thesis the LinearSVC classifier has been used because it resembles the SVM mentioned above within the Scikit-learn module. 10-Fold cross validation is used because it is the best method in which all observations are used for training and testing, and each observation is used for testing exactly once.

4.2 FEATURES

In order for a classifier to work it has to have features it can use to train on. When using machine learning algorithms used to classify text, the data that consists out of text first needs to be prepared before the system can actually use them. The features that are used therefore need to convert the data into numerical data before the algorithm works as it is meant to work. The features used in this thesis will be discussed next.

4.2.1 Bag Of Words

Bag of words (BoW) are the occurrences of words represented as vectors. BoW throws away all information on structure and order of the words in a sentence and focuses solely on their occurrences. For text classification this is usually done by using one of two vectorizers, both included in Scikit-learn (Pedregosa et al., 2011). One way of doing this is by using a count vectorizer that gives every unique word an index number and counts the occurences of these words. This usually leads to good performance, but it can give frequent words more importance than they should have. So another way of doing it is using a term frequency inverted document frequency (tf-idf) vectorizer, where term frequency (tf) measures how often a term occurs in a document and where inverse document frequency (idf) measures how important a term is, giving a higher score to infrequent terms (D Manning et al., 2008). A tf-idf vectorizer also gives every unique word a different index, but uses equation 1 to give words a weight. In the results section, results of both the count vectorizer and the tf-idf vectorizer will be presented.

$$\text{tf-idf}_{t,d} = (1 + \log \text{tf}_{t,d}) \cdot \log \frac{N}{\text{df}_t} \quad (1)$$

When using each of these vectorizers, multiple parameters can be set. The two parameters that have been set for both of the vectorizers in this thesis are stop words and the n-gram range. The stop words that have been used are the stop words included in the NLTK module (Loper and Bird, 2002). This pre-defined stop word list contains commonly used words that don't carry much information. The words contained in this stopwords list can be found in the 'stopwords.txt' file on the GitHub repository of this thesis (Reijngoud, 2018). When using a vectorizer to create vectors out of words, also the use of n-grams can be specified. Looking at n-grams means looking at sequences of words or letters (Johannes, 1999). In this case it means looking at sequences of words in a sentence. Looking at n-grams can improve the results for a text classification like the one in this thesis. The n-gram ranges that have been set for the two vectorizers range from one to three. If the n-gram range is set to one, the vectorizer only includes unigrams, when it is set to two it includes unigrams and bigrams and when it is set to three it includes unigrams, bigrams and trigrams. This is done to construct the most complete classifier.

4.2.2 Difference in Sentence Length

The length of a sentence could be a good indicator of it being a human or a machine translated piece of text. The expectation is that the MT will keep the length of a sentence more similar to its source than a human translator would, because the MT translates sentences per n-grams and the HT translates sentences as a whole. The length of a sentence is the amount of words in a sentence, not the amount of characters. Punctuation is not included in this length calculation. For this feature

it is essential to have the source data provided, otherwise a difference cannot be calculated. The length difference will be calculated by equation 2.

$$lengthdifference = absolute(length(source) - length(target)) \quad (2)$$

4.2.3 Difference in Number of Dots and Commas

Difference in number of dots and commas means splitting the source language up into multiple parts, either by using a comma (",") or a dot (."). As mentioned in Ahrenberg (2017) this feature can give good indications as to whether a sentence is translated by a human or by a machine. Ahrenberg concluded in his research that a human translator sometimes splits a sentence into more parts, in the most extreme case the human translator split a source sentence up into three target sentences. A machine translator system kept the amount of dots and commas more similar to the source. Again for both of these calculations it is essential that the information about the source language is also available, otherwise a difference between source and target language cannot be computed. The difference in the number of dots and commas will be calculated by equations 3 and 4.

$$Difference_{Ncommas} = absolute(source.count(",") - target.count(",")) \quad (3)$$

$$Difference_{Ndots} = absolute(source.count(".") - target.count(".")) \quad (4)$$

4.2.4 Reducing Vocabulary

When all of the above features got combined the classifier ran into a memory error because the array was too large to handle for the 8gb of memory. The vocabulary had to be reduced for the machine to be able to run the combined features classifier. Reducing the vocabulary is done by removing Dutch stop words from the data and excluding words that occur only once. Doing this reduced the vocabulary from 18257 words to 8215 words. This made the classifier able to run with both features combined while it first could not. The results without the reduced vocabulary will be included only for the classifier using BoW. During classification Dutch stop words coming from the NLTK module (Loper and Bird, 2002) have been excluded from the input data. This is done to remove frequent words that hardly contain much information. It reduces noise and it also slightly improves the speed of the classifier.

4.3 EVALUATION

With regard to the research questions, evaluation will be done in two ways. First the classification scores will be presented and next the most informative features will be presented.

4.3.1 Scores

Evaluation of performance on classification tasks is usually done by reporting the precision, recall and F1-score (D Manning et al., 2008). When evaluating a classifier, also the accuracy (represented by equation 5) will be calculated. When evaluating a classifier first a baseline score must be produced for evaluating the performance of the added features from the classifier. Because the labels in this data set are

equally divided, the baseline scores for precision, recall, F1-score and accuracy will be around 0.50.

$$Accuracy = \frac{True\ positives + True\ negatives}{True\ positives + False\ positives + True\ negatives + False\ negatives} \quad (5)$$

When the classifier is built, first the performance on the training data set will be evaluated by reporting the F1-score and accuracy. Then the system must perform on unseen sentences of the test set, this will again be done by reporting the F1-score and accuracy. Because the program is executing a 10-fold cross validation, the mean of these 10 results will be reported.

4.3.2 Most informative features

To answer the research question with regard to the most informative features, the most informative words need to be extracted from either class. Most informative features are words that are most indicative for each of the classes within the classification task.

5 RESULTS

In this chapter the results of all the classifiers as explained in section 4 will be presented. First a table containing all the classifiers and their scores will be shown. Then the most informative features for the different classes in some classifiers will be presented.

5.1 CLASSIFICATION SCORES

r The resulting F1-score and accuracy scores for the different classifiers with its features are presented in Table 3 where features represent the numerical features of difference in sentence length, Ndots and Ncommas and where BoW represents bag of words. The results will be discussed further in 6.1.

Table 3: Results of the classifiers including baseline.

Classifier (features)	F1-score	Accuracy
Baseline (count, Unigrams)	0.50	0.49
Baseline (tf-idf, Unigrams)	0.51	0.49
Len	0.62	0.61
NDots	0.59	0.59
NCommas	0.52	0.60
Ndots, NCommas	0.58	0.62
Len + Ncommas	0.65	0.64
Len + NDots	0.64	0.62
Len + NDots, Ncommas	0.65	0.66
BoW (count, Unigrams, original vocabulary (ov))	0.71	0.71
BoW (tf-idf, Unigrams, ov)	0.73	0.72
BoW (count, Bigrams, ov)	0.72	0.72
BoW (tf-idf, Bigrams, ov)	0.74	0.74
BoW (count, Trigrams, ov)	0.73	0.74
BoW (tf-idf, Trigrams, ov)	0.74	0.74
BoW (count, Unigrams, reduced vocabulary (red))	0.74	0.73
BoW (tf-idf, Unigrams, red)	0.75	0.75
BoW (count, Bigrams, red)	0.73	0.73
BoW (tf-idf, Bigrams, red)	0.75	0.75
BoW (count, Trigrams, red)	0.72	0.71
BoW (tf-idf, Trigrams, red)	0.75	0.75
Len + NDots, Ncommas + BoW (count, Unigrams, red)	0.77	0.76
Len + NDots, Ncommas + BoW (tf-idf, Unigrams, red)	**0.78**	**0.77**

5.2 MOST INFORMATIVE FEATURES

In Tables 4, 5 and 6 the most informative features for both classes for the three best scoring classifiers are presented. The classifier using BoW and trigrams has been excluded as this produced the same results as those using unigrams and birgrams, but it was much slower. The results in the tables will be further discussed in 6.2.f

Table 4: Most informative words for both classes in the classifier with BoW and Unigram.

Score	Human Translation	Machine Translation	Score
-3.6	wij	rapport	2.7
-2.9	dient	autoriteit	2.4
-2.5	eveneens	eur	2.3
-2.4	iedere	diervoeders	2.2
-2.4	**overheidssteun**	ondersteund	2.1
-2.4	tevens	**staatssteun**	2.1
-2.4	vraagstuk	dusverre	2.0
-2.4	wel	verwelkomen	2.0
-2.4	heffing	verwelkom	2.0
-2.3	immers	verheugd	1.9

Table 5: Most informative words for both classes in the classifier with BoW on Unigrams and Bigrams.

Score	Human Translation	Machine Translation	Score
-5.1	wij	rapport	2.8
-3.3	wel	autoriteit	2.4
-3.1	immers	**staatsteun**	2.3
-2.6	vraagstuk	mening	2.3
-2.6	dient	betrekking	2.2
-2.4	daarbij	hebt	2.1
-2.3	**overheidssteun**	lokale	2.0
-2.3	wij moeten	feite	2.0
-2.3	plaatselijke	verwelkom	1.9
-2.2	tenslotte	fabrikant	1.9

Table 6: Most informative words for both classes in the classifier with all features combined.

Score	Human Translation	Machine Translation	Score
-3.3	wijken	rapporteurs	2.7
-2.5	diepe	eurodac	2.7
-2.4	evenredig	autowrakken	2.3
-2.3	medina	diezelfde	2.3
-2.3	plaatsvinden	stabiel	2.2
-2.3	vraagtekens	long	2.0
-2.3	voren	afgenomen	1.9
-2.3	ieren	ondersteunt	1.9
-2.3	tevredenheid	duurt	1.9
-2.2	overige	toegezonden	1.9

6 DISCUSSION

6.1 SCORES

As can be seen from Table 3, the resulting scores for the classifier using only BoW and the scores of the classifier combining the sentence length, Ndots, Ncommas and BoW do not differ that much. Using the classifier with BoW only with unigrams and with bigrams also has no advantages as to only using it with unigrams. The best classifier of them all is the one that combines all features with the tf-idf vectorizer on unigrams. The difference with the classifier using only BoW with the tf-idf vectorizer however is small. It can be said that the sentence length, Ndots and Ncommas feature increased the accuracy of the classifier by 0.02. This is a very small improvement but it means that in some cases in which it couldn't classify correctly using only BoW, these features were good enough to classify into one of the classes. Using the classifier with all features combined with tf-idf on bigrams is not in the results because the array was so large that the machine ran into a memory error. The expectation however is that it would not improve the classifier anyway. This can be concluded from the results in Table 3 where the classifiers with the BoW with the reduced vocabulary do not improve results upon using it on bigrams, as it gives the same results as on unigrams. The reduction of the vocabulary also had a positive effect on the results of the classifiers, as can be seen in Table 3. As noted in Dietterich (1998), statistical significance cannot be calculated well in supervised learning tasks that involve n-fold cross validation, because all available statistical tests have shortcomings when applied to a binary classification task using 10-fold cross validation.

6.2 MOST INFORMATIVE FEATURES

The most informative features shown in Tables 4, 5 and 6 show the most important words for each of the classes. For the BoW-only classifier the most informative words for both classes do have some words in common. When looking at Tables 4 and 5 a nice differentiation between HT and MT can be found in the word 'overheidssteun' for HT and the word 'staatssteun' for MT (presented in bold), these words translate to the English 'state aid' in the data. These words reference to the same word in the OG sentences but apparently HT uses 'overheidssteun' more often and MT uses 'staatssteun' more often. Another example are the words 'plaatselijke' for HT and 'lokale' for MT (underlined in Table 5), both referring to the word 'local' in the OG data. These again are synonyms to each other but HT and MT seem to have their preference for using one or the other.

The classifier using all features combined, extracts different most informative features (Table 6) then already presented in Tables 4 and 5. This is remarkable because one would expect that the most informative features wouldn't change much by adding three columns of numerical features. When looking at the data in how many sentences the words presented in Table 6 occur, it's mostly words that occur almost evenly across both HT and MT. This means that the added numerical features had impact on the classification task and not so much the BoW. It would be interesting to see a most informative features table for the numerical features too, but the implementation of that was problematic.

7 CONCLUSION

The classifier that was created for this thesis is indeed able to classify texts as either HT or MT. It did that with an accuracy score of 0.77. This, compared to the baseline score, is good. So it can be concluded that it is possible to create a classifier that can successfully classify a text as HT or MT. The most informative features for the best scoring classifier were the words as presented in Table 6. The added features of sentence length, Ndots and Ncommas also had effect on the accuracy score of the classifier. Without those features the classifier had an accuracy of 0.75, which means those features improved the accuracy with 0.02, which on the 19790 sentences is quite an improvement.

Reducing the vocabulary by discarding words that appear only once was a good choice, because it made the classifier run faster and it also improved results for the classifiers. Without reducing the vocabulary, the classifier with combined features would not be able to run.

Limitations of this thesis research were that the classification task was conducted on a data set that was aligned over the classes and that had the source language available. In practice this wouldn't be useful as a classifier to find fraudulent websites because in that case you wouldn't have the source language available and you wouldn't have enough data from human translations to train on. The data set in this thesis was also very domain specific, namely the European parliament. For a broader view on most informative words for HT or MT a less domain specific data set would be better.

Another limitation that only became evident during testing, was that combining the BoW with the other features produced such a large array that the system ran into memory errors when trying to classify using more then just unigrams. This is why there are only unigram scores reported for the classifier that combined all features. This could be resolved by using a computer system with more memory than 8gb.

BIBLIOGRAPHY

Ahrenberg, L. (2017, Nov). Comparing machine translation and human translation: A case study.

Bentivogli, L., A. Bisazza, M. Cettolo, and M. Federico (2016). Neural versus phrase-based machine translation quality: a case study. *CoRR abs/1608.04631*.

Cortes, C. and V. Vapnik (1995, Sep). Support-vector networks. *Machine Learning 20*(3), 273–297.

D Manning, C., P. Raghavan, H. Schtextbackslashütze, and E. Corporation (2008, 01). *Introduction to Information Retrieval*, Volume 13, pp. 164–165.

Dietterich, T. G. (1998). Approximate statistical tests for comparing supervised classification learning algorithms. *Neural computation 10*(7), 1895–1923.

Johannes, F. (1999, 02). A study using n-gram features for text categorization.

Koehn, P. (2005). Europarl: A Parallel Corpus for Statistical Machine Translation.

Kumar Yogi, K., C. Kumar Jha, and S. Dixit (2015, Dec). Classification of machine translation outputs using nb classifier and svm for post-editing. *2*, 21–29.

Li, Y., R. Wang, and H. Zhao (2015). A machine learning method to distinguish machine translation from human translation. In *Proceedings of the 29th Pacific Asia Conference on Language, Information and Computation: Posters*, pp. 354–360.

Loper, E. and S. Bird (2002). Nltk: The natural language toolkit. In *Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics - Volume 1*, ETMTNLP '02, Stroudsburg, PA, USA, pp. 63–70. Association for Computational Linguistics.

Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al. (2011). Scikit-learn: Machine learning in python. *Journal of machine learning research 12*(Oct), 2825–2830.

Refaeilzadeh, P., L. Tang, and H. Liu (2009, 01). Cross-validation. *532–538*, 532–538.

Reijngoud, W. (2018). Information science bachelor thesis 2018. https://github.com/wesselreijngoud/THESIS2018.

Susik, M. and S. Han (2015). Py-googletrans 1.2. https://github.com/ssut/py-googletrans.