INTERNATIONAL HELLENIC UNIVERSITY

Confidence score estimation for English to Greek machine translation

Orfanoudakis Dimitrios

SID: 330817001319

SCHOOL OF SCIENCE & TECHNOLOGY

A thesis submitted for the degree of

Master of Science (MSc) in Data Science

—

FEBRUARY 2020

THESSALONIKI – GREECE

INTERNATIONAL HELLENIC UNIVERSITY

Confidence score estimation for English to Greek machine translation

Orfanoudakis Dimitrios

SID: 330817001319

Supervisor: Prof. Konstantinos Diamantaras

SCHOOL OF SCIENCE & TECHNOLOGY

A thesis submitted for the degree of

Master of Science (MSc) in Data Science

FEBRUARY 2020

THESSALONIKI – GREECE

Abstract

This dissertation was written as a part of the MSc in Data Science at the International Hellenic University. NLP applications often use text-to-text transformations, in which a system given a natural language word sequence as input, is expected to generate an alternative version of this text as output, also in natural language. In Machine Translation, the evaluation of this output can be done in two ways. Firstly, by comparing the MT output to one or more reference outputs with the help of distance-based evaluation metrics and secondly, by building ML models, trained on large human-annotated datasets, that aim at predicting the quality of MT outputs when reference translations are not known. Following the second approach, the goal of this dissertation is to develop a Quality Estimation (QE) model able to predict confidence scores for given English to Greek automated translations. For that, several machine learning algorithms are explored and trained on a dataset of 77720 human-annotated English to Greek translation tuples, where each of these tuples consists of the source, the target and the edited segment.

I would first like to thank my thesis supervisor Professor Konstantinos Diamantaras for giving me the chance to elaborate on such an interesting field of research and TAUS, for providing me with this dataset and the continuous support. My deepest gratitude goes to my family, friends, and colleagues for their support, encouragement, and patience throughout all the years of my studies.

Orfanoudakis Dimitrios
02/02/2019

Contents

1 Introduction

1.1 Problem Statement

In the era of AI, where various NLP applications aim to produce high quality natural language output, the task of evaluating such systems has become mandatory defining a new topic of research. In Machine Translation, where the goal of systems is to produce outputs as close as possible to human-like translations, the progress over the past years has been tremendous and systems especially in Statistical Machine Translation (SMT) and in Neural Machine Translation (which is currently the state-of-the-art in the field), appear to reach impressive breakthroughs. Despite their high performance, these systems are not yet in a position to deliver highly trustable output making the task of post-editing, and therefore the human factor, a necessary step before publishing an output. Post-editing is usually performed by a human annotator (HT), a translator or a linguist, who reviews a translation output generated by an MT system and corrects it where needed to make it intelligible. This task can become very costly (timewise but not only) and usually requires even more time than the translation process itself, especially in cases where reviewers have little or no knowledge of the source language or in cases where the load of translations to be judged keeps increasing and instant quality-responses are needed. Considering that as a motivation, several different methods have been developed aiming at minimizing human effort during the post-editing phase. One of them, and probably the most important, is widely known under the name of Confidence Estimation (CE) or Quality Estimation (QE). The idea behind QE is to build a machine learning model that is capable of pointing out the correct parts, detecting translation errors, concluding the overall quality and providing a confidence score for each MT hypothesis without having access to human-reference translations (Blatz et al., 2004). Such an indication can improve accuracy and reduce considerably the time needed for post-editing, as it reveals whether a specific translation is worthy of being corrected or not (needs less time to be corrected than to reproduce another one from scratch) redrawing post-editors' focus to the sentences that likely needs editing. It is important to distinguish the difference between the terms Estimation and Evaluation. In Quality Evaluation the system compares MT outputs with their corresponding reference translations and turns differences and similarities into a score that indicates the quality of each output, such as BLEU (BiLingual Understudy) (Papineni et al., n.d.), NIST (Martin & Przybocki, n.d.). The key differentiator between these two approaches is that quality estimation does not require a human reference translation.

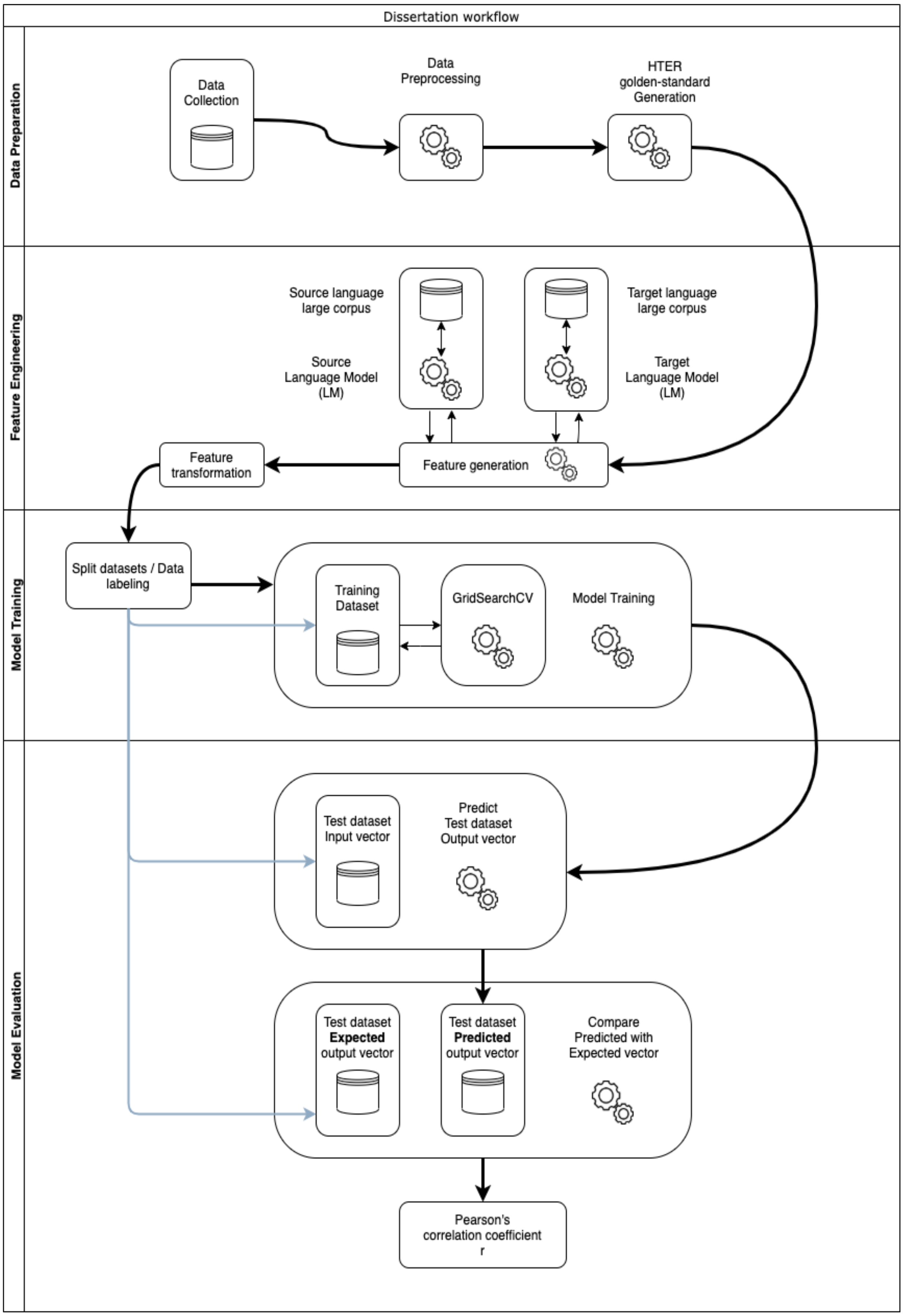
1.2 Thesis Objectives

The aim of this thesis is to develop a machine learning model capable of predicting the quality of MT system outputs without the need of reference texts. To achieve that, a pipeline-system is built that handles all the required steps that take place during this supervised-learning ML task.
More specifically the objectives to achieve this are as follows:

- Thorough research regarding Machine Translation Quality Estimation. Why QE is needed, and how it can be applied. Investigation of common and latest models and frameworks.
- The collection of a relatively large corpus of bilingual data from English to Greek. This dataset is provided by TAUS[1] a language technology company and it consists of approximately 77720 segment-pairs, each one with its corresponding reference post-edited translation text by a human translator (HT).
- The preprocessing of the collected dataset using several different manual cleaning methods and tools including Bicleaner[2] and Bifixer[3]. After this stage, the dataset ends up with a significantly smaller size of approximately 16000 data points.
- Computing HTER (Human-targeted Translation Edit Rate) (Matthew Snover et al., n.d.) scores between each MT translation and the corresponding post-edited version, using a tool written in Java named TERcom[4] (Matthew Snover et al., n.d.). HTER score is considered as the "golden standard" and everything is calculated with respect to that.
- The training of both source and target language models on large corpora, necessary for generating n-gram and POS-tagged n-gram language model features.

[1] https://www.taus.net/
[2] https://github.com/bitextor/bicleaner
[3] https://github.com/bitextor/bifixer
[4] http://www.cs.umd.edu/~snover/tercom/

- The feature engineering process where the main goal is to transform the previously cleaned dataset into a form easier to interpret by algorithms, and often by people without a data-related background (mainly through data visualizations). To generate meaningful features, various techniques were applied including dimensionality reduction, log-transformations, categorical grouping, data-scaling and more.
- Implementation of several ML algorithms in an attempt to address the sentence-level QE confidence score prediction as a supervised machine learning regression task. Support Vector Machine, Random Forest, eXtreme Gradient Boosting and Multilayer Perceptron feedforward neural network were deployed to tackle the task.
- Evaluation and comparison of the models with respect to Pearson's correlation coefficient r.


Dissertation workflow
Data Preparation
Data Collection
Data Preprocessing
HTER golden-standard Generation
Feature Engineering
Source language large corpus
Source Language Model (LM)
Target language large corpus
Target Language Model (LM)
Feature transformation
Feature generation
Model Training
Split datasets / Data labeling
Training Dataset
GridSearchCV
Model Training
Model Evaluation
Test dataset Input vector
Predict Test dataset Output vector
Test dataset Expected output vector
Test dataset Predicted output vector
Compare Predicted with Expected vector
Pearson's correlation coefficient r

Picture 1: Dissertation's implementation workflow

1.3 Thesis Composition

The rest of the dissertation is formed as follows:

- In Chapter 2, the relevant literature and the latest developments in the field of Machine Translation Quality Estimation are summarized and the background theory, which is necessary for the comprehension of the problem, is outlined.
- In Chapter 3, the data and methods used in this thesis are analyzed. Acquisition of data and data preprocessing methods are introduced, and the process of feature extraction is described.
- In Chapter 4, the implementation of several shallow machine-learning algorithms is performed. The results obtained from testing the trained model on the selected dataset, are then presented.
- In Chapter 5, the conclusions are summarized, and the direction of future studies is suggested.

2 Literature Review

2.1 Machine Translation

2.1.1 Overview

Globalization

In the era of globalization, where the interdependence of the world's economies, cultures, and populations is constantly growing, a new landscape of challenges has already been introduced. Over the years, serious efforts have been made towards the path of facilitating an interconnected world that serves and benefits the masses. These efforts, which are directly or indirectly related to human communication, affect multiple-different contexts such as international trade, culture, media, products, public services and have a common mission, to reduce barriers allowing the emergence of an international network of economic systems. Reducing the factor of time, distance and making information more easily accessible than ever, the last remaining barrier to overcome while moving from local to global, as the term globalization suggests, is language.

Machine Translation

The ever-increasing need for inter-regional communication and language translation in recent years has made translation nowadays a key mediator of global communication. Every day, soaring flows of information in the form of text, including scientific and technical reports, legal documents and more, are requested to be translated. However, traditional translation, due to its tedious and repetitive nature that requires consistency and precision, can become a very time-consuming task that is unable to meet this growing demand. Thus, MT has become the preferred method of translating content.

Although its effectiveness, MT still faces numerous challenges some of which are listed below (Garg & Agarwal, 2018):

- Not every word in source language has an equivalent word in the target language
- Words in a language can have a number of meanings
- Differences in the grammatical rules and syntax of source and target languages

Aiming at solving these issues, among others, different approaches have been introduced over the last fifty years from MT researchers who will to increase the quality of MT and build robust MT systems that perform well on different language pairs.
In the following section, the main MT architectures are described in more detail.

2.1.2 MT architectures

Rule-Based Machine Translation (RBMT)

The first actual implementation of machine translation was RBMT developed many decades ago. The idea is, to first tokenize a source sentence into words trying to identify their meaning and then, to map them into tokens of the target language based on a set of rules defined manually by linguists. These rules are designed to map the relationships between the structure of the source and the target language. One of the upsides of RBMT is that in the case of a well-developed system, a wide scope of content can be translated without the requirement of huge bilingual corpora, as in SMT. However, the process of building an RBMT framework only for one language-pair may take several-years as it is intensive and time-consuming work. Moreover, human-defined rules are not able to cover perfectly all conceivable linguistic phenomena, which may result in bad quality outputs when facing real-world input texts. For instance, it has been shown that RBMT does not perform satisfactorily in cases of slang or metaphors. Nowadays, most of rule-based MT systems have been replaced by SMT or Hybrid architectures to a great extent. Nevertheless, in cases of less common languages where training data is limited Hybrid systems are preferred since SMT systems require very large bilingual corpora to be trained.

Statistical Machine Translation (SMT)

Statistical machine translation models are trained on an enormous size of high-quality bilingual and monolingual corpora. The idea that an SMT engine follows is to search for statistical correlations between source and target texts, looking both for whole segments and smaller text sequences inside each segment as it builds the model. There is no usage of grammar or any rule by the model as in RBMT, instead, it generates confidence-scores that represent the possibility of a source sentence to map to a specific translation text. SMT is one of the most commonly used MT architectures in the industry. Popular systems like Google Translate and Bing Translator are using SMT in their implementation. Despite that with a large enough training corpora you can train generic language-pair independent translation models, the main weakness of SMT is that it requires enormous and efficient bilingual corpora to be properly trained. Also, SMT performs poorly when it is required to translate texts that are not similar to already seen training corpora. For instance, a system trained with texts coming from a technical domain will face problems trying to translate texts coming from a different domain. In this manner, it is essential that models are trained on data similar to the texts that later will be requested to translate.

Example-Based Machine Translation (EBMT)

EBMT engines employ translation by analogy. To do so, the system looks for existing identical translation pairs of source and target sentence examples. When a new input text comes in, the system looks for examples that are identical in their source text. After it finds the required examples, the target sentence is generated by imitating the translation-part of the previously matched examples. And conversely, when no similar examples are found, the translation quality may be very poor. Because of this uncertainty, EBMT was not widely adopted in the industry.

Neural Machine Translation (NMT)

NMT uses neural networks comprised of nodes that are designed inspired by the human brain architecture. Each node represents a single word, a sentence, or a longer segment that interacts with another node in a framework of complex relationships where bilingual texts are used to train the network. The design complexity of these networks allows the meaning of any word to be transformed into considerably more educated guesses about its context. Neural Machine Translation systems are continuously being adjusted during the training process. NMT systems require a lot of processing power and their training can become a very computationally intensive task. This is the reason why this type of MT systems has become feasible only in recent years, in which the latest developments in the field of hardware have introduced new solutions able to address such computationally intensive tasks.

Hybrid Machine Translation

Hybrid machine translation systems combine multiple MT approaches in a single translation system, as all of the previously mentioned systems have their shortcomings Hybrid systems are divided into two categories. The first consists of rule-based engines that employ SMT for post-processing and cleanup whereas, the second category consists of SMT systems guided by rule-based engines. Either of the above is used with some input from an NMT system. In the first scenario, after the source segment is translated from an RBMT system, it is processed by an SMT engine that searches for any errors. In the second case, the RBMT engine aims simply at inserting metadata (e.g. noun/verb/adjective, present/past tense, etc.) as a support to the SMT engine. To a certain extent, all real-world MT systems adopt hybrid systems that combine rule-based and statistical approaches.

2.1.3 Computer Assisted Translation tools (CAT tools)

CAT tools (Computer-Assisted Translation tools) are computer programs employed by professional translators, linguists, and language service providers in order to improve the quality of their work. They are designed to help users translate faster and more efficiently documents from one language to another. A CAT tool typically divides texts into sentences or even smaller pieces and shows them in an easily understandable way to the user. Segments are usually displayed in a special box, close to which there is a translation editor where users are allowed to enter a translation or modify a machine translation suggestion. Once a translation is approved and submitted, alongside its source segment, it is treated as a translation unit and is stored for later use as translation memory. Translation memory (TM), among others, is an essential feature of a CAT tool. Its mission is to store translation units after they are created into a database so that they can be reused during the translation of related new texts. Apart from perfect matches, segments that do not match 100% can also be detected and retrieved using special "fuzzy-search" features. Allowing previously processed, related translation-parts to be reused, Translation Memory saves time and helps the translators to use consistent terminology. Another important feature that most of CAT tools now support is the interactive Machine Translation. This feature aims at predicting the translation that a human is about to enter by providing a number of translations. These suggestions can refer either to a part or to the whole sentence to be translated. There are more automation tools that a CAT tool comes with, including automatic translation following glossaries and quality checks, dynamic machine learning, spell checkers and other translation automation tools. It is estimated that nowadays over 85% of translators take advantage of CAT tools to improve their productivity.

2.1.4 Post Editing for Machine Translation

Regardless of the automation that CAT systems provide to users, the human factor, in the role of post-editors, is still required during the translation process as in many cases, quality levels of MT output are still far from human standards (HT). According to the "Post-editing in Practice" report[5] by TAUS, "Postediting is the process of improving a machine-generated translation with a minimum of manual labor. The result will be either a publishable document (full post-editing) comparable to high-quality human translation or an understandable document (light post-editing), containing correct terminology and names, expressed in unambiguous but not necessarily elegant sentences". Post Editing for Machine Translation combines the best of both human quality and machine efficiency, and this is why PEMT over the last years has become an integral part of almost every workflow in the translation industry. At the same time, the smooth adaptation of the PEMT practice by the industry has created the need for organizations to be able to effortlessly train and review their post-editors. Given that there is a number of different PE methodologies in use nowadays, organizations adopt different approaches to evaluate performance. There are systems that evaluate PE performance counting only on post-editor's productivity, others that mark the output as "over-edit" or "under-edit" based on the post-editor's effort and there are systems that measure the percentage of the accepted MT hypotheses against the rejected MT hypotheses in the final output. Most of them, are mainly distinguished by quality and time-spent due to the customers' expectations and needs, but other characteristics such as turnaround time, MT acceptability and costs are also considered.

[5] https://www.taus.net/think-tank/reports/postedit-reports/postediting-in-practice

2.2 Workshop for Machine Translation (WMT)

The Conference on Machine Translation (WMT) provides a venue, annually since 2006, for researchers from the area of machine translation to compete on various MT shared tasks and build state-of-the-art systems. To do so, participants are able to use, rare and valuable data sets, provided by the competition to experiment and facilitate further research. Its main focus is to gather both academic and commercial researchers in a well-defined and controlled framework that allows the evaluation and consequently the improvement of MT technologies. In the past and before it is presented as a conference, WMT was being held as a workshop for ten years from 2006 to 2015 under the name of Workshop on Statistical Machine Translation. However, because of the movement from Statistical to Neural Machine Translation, the name of the conference changed to Conference on Machine Translation. The WMT shared tasks are open to every academic and commercial research lab across the world, to compete and present their own research papers. Nowadays, the WMT campaign consists of 8 shared tasks. The most popular of them, are the machine translation tasks, where contestants test their MT system on a shared test set, in any of the 18 language pairs, trying to reach the best performance. In all MT tasks, systems are evaluated in terms of the human judgment of translation quality. There are 4 MT shared tasks, the machine translation of news and the similar language translation (Barrault et al., 2019), the machine translation robustness (Li et al., 2019) and the biomedical translation (Bawden et al., 2019). Apart from the MT tasks the competition includes also, the Automatic Post-Editing (Chatterjee et al., 2019) task that aims at developing methods used to automatically identify and fix errors that are generated by unknown MT engines, and the Parallel Corpus Filtering (Koehn et al., 2019) task where the goal is to develop methods able to automatically extract good quality sentence pairs from a given noisy bilingual corpus. (*2019 Fourth Conference on Machine Translation (WMT19)*, n.d.). In addition, WMT includes 2 more shared tasks on evaluation. The first task is focusing on MT automatic evaluation metrics. The main goal of this task is to find the automatic metric that achieves the strongest correlation with human judgments of MT quality. For that task, participants have access both on the target and reference translations. The second task of the evaluation shared tasks is the MT quality estimation. In this task, participants are required to develop systems that are able to estimate how good MT translation output is, without having access to reference translations. This task offers an analysis of the current evaluation methodologies applied widely within the quality estimation landscape.

2.3 MT Quality Assessment

Because of the rapid development of MT technology and the continuous efforts for progress towards different directions, the need for benchmarking MT systems' performance has become necessary. However, MT quality evaluation can become a very challenging task. Natural languages are often ambiguous and in many cases, the same content in different languages is not reflected in a similar fashion (Arnold, 2003). MT quality evaluation is not only useful for benchmarking newly developed MT systems, it is also a very important task for users during the translation process of unseen data. For many years, translators were responsible for judging whether a translation was good or not, given a source text and its translation. Nevertheless, the task of MT quality evaluation by humans turned out to be very costly in terms of time, and in some cases, where users did not have enough experience in the source or target language, it was not even possible. As a result, automatic methods for evaluating the quality of MT systems has become a real need. These methods are applied either by filtering out low-quality segment pairs to prevent translators from spending time on bad translations or by presenting translations in a way that the level of their quality is known to the end-users. Depending on their application and purpose, these methods can be easily distinguished on the ones that use reference translations, known as MT Evaluation metrics and the ones that do not, known as MT Quality Estimation systems. The design of automatic MT evaluation metrics is a hard task due to the multiple possibly acceptable reference translations for each source text. These systems need to be adaptable and flexible enough allowing variations in translation and also be able to identify and penalize oddities and deviations. Furthermore, a strong correlation with human judgments is important for their reliability.

2.4 Evaluating MT Evaluation systems

The most widely used method for measuring the distance between the automatic evaluation predictions of a system and their corresponding human judgments (golden standards), is the correlation coefficient r. A correlation coefficient measures how similarly two variables change, as well as the direction and strength of this relationship. It is also employed in multiple tasks of WMT workshops. Two most popular types of correlation metrics are Pearson and Spearman rank correlation:

Pearson Correlation

Pearson's correlation coefficient ρ (Pearson, 1895), evaluates the linear relationship between two continuous variables. It describes how a change in one variable is associated with a proportional change in the other. Given two random variables X and Y, the correlation is calculated as follows (Montgomery & Runger, 1994).

$$\rho_{XY} = \frac{cov(X,Y)}{\sqrt{V(X)V(Y)}} = \frac{\sigma_{XY}}{\sigma_X \sigma_Y}$$

On a set of paired data (X, Y) as (x_i , y_i) and i = 1..n, the Pearson correlation coefficient is described as:

$$\rho_{XY} = \frac{\sum_{i=1}^{n}(x_i - \mu_x)(y_i - \mu_y)}{\sqrt{\sum_{i=1}^{n}(x_i - \mu_x)^2}\sqrt{\sum_{i=1}^{n}(y_i - \mu_y)^2}}$$

where μ_x and μ_y are the means of discrete random variable X and Y.

Spearman rank Correlation

Spearman rank correlation coefficient is a simplified alternative to Pearson's correlation coefficient used also for measuring how correlated automatic evaluations and golden standards are.

Provided there are no ties, Spearman's rank correlation coefficient (r_s) is described as follows:

$$rs_{\varphi(XY)} = 1 - \frac{6\sum_{i=1}^{n} d_i^2}{n(n^2-1)}$$

Where d_i gives the difference between the two rank variables $(x_i - y_i)$

for $X = \{x_1, x_2, ..., x_n\}$ and $Y = \{y_1, y_2, ..., y_n\}$ that describes the system ϕ.

In the case of tied ranks, the Spearman's rank correlation coefficient is:

$$rs_{\varphi(XY)} = \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_i (x_i - \bar{x})^2 \sum_i (y_i - \bar{y})^2}}$$

Mean Absolute Error (MAE), Root of Mean Squared Error (RMSE)

Two of the most widely used evaluation metrics for measuring performance on regression are employed in the QE scoring:

- Mean Absolute Error (MAE)
- The Root of Mean Squared Error (RMSE)

On a set of observations S, where $1 < i < |S|$ and $N=|S|$, $H(s_i)$ is the predicted score of an item s_i and $V(s_i)$ stamds for the expected value of item s_i.

$$\text{MAE} = \frac{\sum_{i-1}^{N} |H(s_i) - V(s_i)|}{N}$$

$$\text{RMSE} = \sqrt{\frac{\sum_{i-1}^{N} (H(s_i) - V(s_i))^2}{N}}$$

MAE and RMSE are automatic, nonparametric and deterministic metrics.

2.5 MT Reference-based Evaluation

2.5.1 Automatic Evaluation Metrics

Since human evaluation faces some weaknesses being time-consuming, not reproducible and expensive, automatic evaluation metrics are the preferred way of evaluating machine translation. The general goal of MT Evaluation is to compare a machine translation to its corresponding human reference translation(s) providing a score that indicates how close the texts are (L Specia et al., n.d.). Therefore, quality depends on human-likeness, although this claim is not accepted by all (Albrecht & Hwa, 2007). Typically, there are two categories of classifying Automatic MT evaluation metrics. There are metrics that rely on Lexical features and metrics that focus more on Linguistic features. Although they differ a lot, in some cases, it is difficult to separate them clearly as they may integrate with each other (e.g. there are lexical metrics that also use certain linguistic features). Another differentiation is, on how they account for reordering and synonyms. Both metric categories are analyzed in the following section.

2.5.2 Lexical Features

Common Lexical evaluation metrics measure the word order, n-gram overlapping, the number of words and word sequences in common, and the edit-distance. Some of the advantages of those metrics are, that they are good at detecting the translation fluency (Han, 2016) and that they are low cost and very fast. On the other hand, there are also few disadvantages as the syntactic information is rarely considered and lexical similarity does not necessarily reflect the similarity in meaning. Some metrics from this category that focus on edit-distance are:

- WER (Tillmann et al., 1997; Vidal, 1997)

  Word Error Rate (WER) metric calculates the minimum number of edits (insertion, deletion, and replacement of a word by another) that must be performed on the machine translation to be identical to the reference translation. It takes word order into account, and in case of a "wrong", according to reference translation, word order in translation hypothesis the WER metric can become significantly low.

$$\text{WER} = \frac{1}{N_{ref}} \sum_{k=1}^{K} \min_{r} \{d_L(ref_{k,r}, hyp_k)\}$$

where $d_L(ref_{k,r}, hyp_k)$ stands for the Levenshtein distance between reference and hypothesis texts.

- PER (Tillmann et al., 1997)

  Position-independent word Error Rate (PER) measures the difference in the number of times that identical words appear in hypothesis and reference sentences, normalized by the number of words in the reference. As the name suggests PER doesn't take word order into account.

$$\text{PER} = \frac{1}{N_{ref}} \sum_{k=1}^{K} \min_{r} \{d_{\text{PER}}(ref_{k,r}, hyp_k)\}$$

  and

$$d_{\text{PER}}(ref_{k,r}, hyp_k) = \frac{1}{2}\left(|N_{ref_{k,r}} - N_{hyp_k}| + \sum_{e} |n(e, ref_{k,r}) - n(e, hyp_k)|\right)$$

  where

  - $n(e, hyp_k)$ is the number of times the word e appeared in hypothesis
  - $n(e, ref_{k,r})$ is the number of times the word e appeared in reference

- TER (Matthew Snover et al., n.d.)

  Translation Error Rate looks for the minimum number of edits required to transform a hypothesis into a reference translation. These edit operations are the deletion, the insertion, the word substitution, and word-sequence shifts. The difference between TER and WER metrics is that the latest is not taking into account the shift operation. Shifts avoid the excessive penalization of word-sequence reordering. When the lowest number of operations is measured, TER is computed by dividing the number of edit operations to the number of tokens of the reference translation, or the average number of tokens of reference translations if a number of references are available.

$$\text{TER} = \frac{\text{\# of edits}}{\text{average \# of reference words}}$$

- TERp (M. Snover et al., 2008)

TER-Plus follows the idea of TER and additionally incorporates three more operations in order to optimize the correlation between the metric scores and human judgments. The word stem matches, the multiword matches using a table of scored paraphrases and the WordNet synonym matches. In that way, instead of aligning only exact matches of words between the hypothesis and target reference, TERp aligns words also when they have a stem in common or are synonyms

- HTER

Human-mediated Translation Error Rate is a variant of the TER metric that is also used on WMT QE evaluation tasks. Instead of just measuring the edit distance of a machine translation hypothesis to its corresponding target, as TER does, HTER first requires human factor in selecting the best reference out of a list of fluent references that are provided by the system.

- BLEU (Papineni et al., n.d.)

BiLingual Evaluation Understudy aims at evaluating the quality of a machine translation by measuring the percentage of n-grams that appear in both the machine translations and the reference translations. The final score is generated by using the n-gram precisions for n = 1..4 in a geometric mean, multiplied by a brevity penalty that comes from cases of machine translations being shorter than the references sentences as seen in the equation below:

$$\text{BP} = \begin{cases} 1 & \text{if } c > r, \\ e^{1-\frac{r}{c}} & \text{if } c <= r. \end{cases}$$

The following equation then

$$\text{BLEU:}N = \left(\prod_{n=1}^{N} \frac{n\text{-grams}(T \bigcap R)}{n\text{-grams}(T)} \right)^{\frac{1}{N}} BP$$

More metrics that were developed on similar logic are:

- NIST (Doddington, 2002)
- ROUGE (Lin, 2004)
- GTM (Turian et al., 2003)
- METEOR (Lavie & Agarwal, 2007)

2.5.3 Linguistic Features

In contrast to the previously mentioned methods that emphasize mainly on lexical features, most recently developed metrics focus on linguistic information, like syntax and semantics, by taking into consideration features like part-of-speech tags, synonyms, textual entailment (TE), sentence structure, paraphrase, named entities, semantic roles and more. Some examples of these metrics are:

- ULC (Giménez & Màrquez, 2008)

ULC is a linear combination of standard lexical metrics, of equal importance, with syntactic and semantic information.

- ROSE (Song & Cohn, 2011)

ROSE combines n-gram and linguistic features using supervised ML techniques. It also uses recall and precision for different size lexical n-grams and information on punctuation, content and part-of-speech (PoS).

2.6 MT Reference-free Quality Estimation

The task of MT Quality Estimation aims at predicting the quality of a system's output for a given source text without any human intervention. To do so, models need to be trained on a large dataset of already annotated source-target sentence pairs with a size that depends on its quality and the selected ML algorithm. While a few thousand examples are usually enough, larger numbers of train datasets lead to higher quality models.
QE covers multiple different levels of prediction, although most of the current efforts aim at the Word-level, Sentence-Level, and Document-level. QE is not intended to estimate the overall performance of MT systems, but rather to estimate if an individual MT unit is worth to be manually corrected (PE). For example, QE applied in a number of sentences to find which of them should be post-edited by ranking them, and QE applied for finding words to be reviewed by the post-editor.

2.6.1 Motivation

Reference-free MT assessment approaches were developed as an attempt for real-world applications to overcome the dependency of automated evaluation metrics on human references. At the early stages of QE, the idea was to predict how confident a particular system was regarding its output (Confidence Estimation), taking into account its system-dependent, glass-box, features. This is not the case any-more as the focus now, is on any helpful quality feature that a system can offer, regardless of the type of features in use (glass-box or black-box). Influenced by this idea, most works nowadays focus on the black box only features as they seem more beneficial for a number of reasons. Black-box features are typically generated from the source and target text such as number of commas, sentence length, n-gram LM probabilities, and POS tags, whereas, glass-box refers to the internal features of the MT engine that were produced the translation such as SMT model score, hypothesis scores and n-best lists (L Specia et al., n.d.). Black-box features handle the assessment of translations without being dependent on the access of the internal components of an MT system. This leads to a less computationally costly assessment solution, which is preferable, especially in cases of commercial systems. Additionally, system-independent features are able to evaluate translations that are coming from any MT system, regardless of the type of the system (rule-based, statistical, hybrid, NMT, etc.). This idea has allowed researchers to focus on how good a translation is on its own, using black-box features, and experiment with features from different sources.

2.6.2 Applications

There are several examples of Quality Estimation for Machine Translation being applied for different purposes including:

- Minimizing post-editing effort by filtering out segments with bad quality as they need more time to be corrected than translating from scratch
- Rerank candidate translation hypotheses produced by an MT system, using another method to help choosing the best candidate (Luong et al., 2015)
- Publish good-quality translations as they are, without the need of being post-edited (Soricut & Echihabi, 2010)
- Selecting a translation from either an MT system or a TM for post-editing (He et al., 2010)
- Selecting the best translation from multiple MT systems (Avramidis, 2013)
- Estimate post-editing effort and time
- Highlighting subsegments that need to be reviewed (Bach et al., 2011)

In what follows, some of the latest developments that outperform in the field of QE are presented.

2.6.3 Shallow feature-engineered QE methods

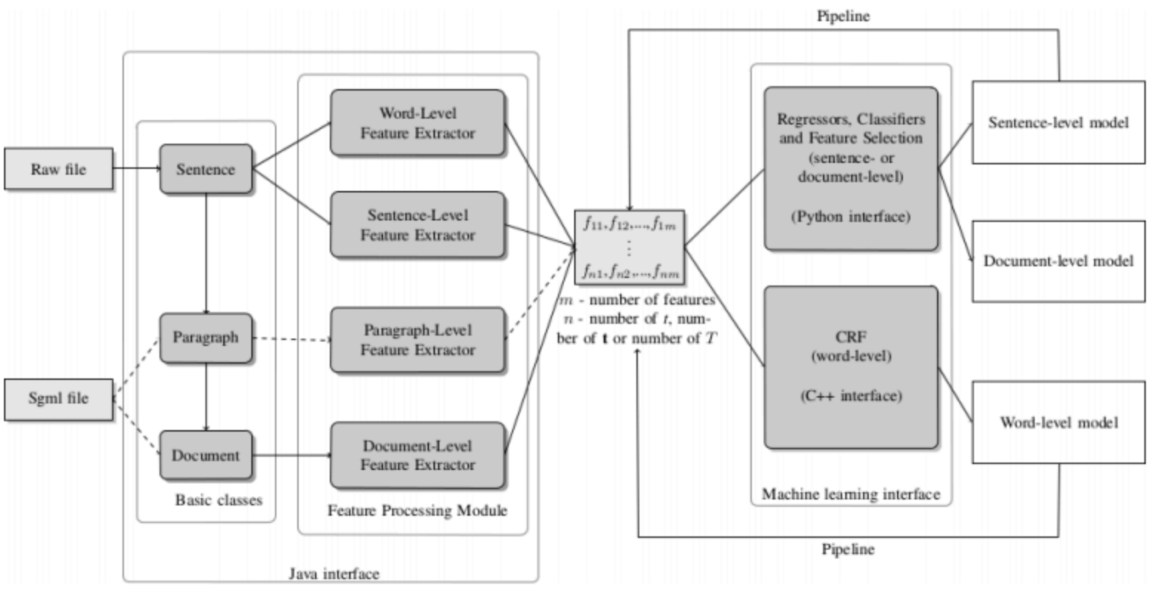
Traditional approaches consist of two main parts. The first one employs feature engineering technics to extract a list of features out of the source and target sentences, that explain the fluency, the adequacy, and the complexity of the translation using external language resources. The second is focusing on building the machine learning model that is finally going to predict the quality of the translations. While simplistic features including language model scores, token counts and punctuation counts are relatively easy to extract, feature engineering can become very cumbersome when it comes to extracting more sophisticated and valuable features. In the case of document, paragraph or sentence levels, QE is approached as a supervised regression problem using algorithms such as Support Vector Machines (SVMs) and Multilayer Perceptron to predict automatic scores (e.g., BLEU, HTER, etc.). Whereas in the case of word-level and phrase-level, QE is approached as a classification task and algorithms such as Conditional Random Fields (CRFs) and Random Forests are employed to classify data points to OK or BAD.

QUEST++ (Lucia Specia et al., 2015)

One of the most widely used implementations on MT QE is the QuEst++ framework. Its development took place at the University of Sheffield by Lucia Specia and her team, and it includes a number of contributions by the research community. It is an open-source tool that supports QE for MT at word, sentence and document level. It provides pipelined processing, meaning that predictions made at a lower level (e.g. for words) can be used as input while training models at a higher level (e.g. sentences). QUEST++ enables a variety of features to be extracted and supports different machine learning algorithms for building and evaluating QE models. For a number of years and in certain language-pairs, QUEST++ has been shown to achieve state-of-the-art performance. Most recent work focuses on sentence-level QE. This variant approaches the task as supervised machine learning using different algorithms to build models out of a large dataset with annotated (e.g. 1-5 likert scores) sentence translation pairs. Sentence-level QE has been one of the main WMT shared tasks, annually since 2012. While standard algorithms can be used to build prediction models, the key to this task is the work of feature engineering. Quest++ consists of two main modules, the feature extractor and a module for machine learning. For many years, the feature-extractor module was used as the official baseline set of features for the QE task in WMT shared tasks. The basic requirements for the feature extraction module to function, are the raw source and translation text files, and a few resources (where available) such as the MT source training corpus and source and target language models. The features in QUEST++ depending on the language pair range from 80 to 123. Some examples of these features are:

- number of tokens in s & t and their ratio
- Source & target LM probability
- Source & target punctuation symbols ratio
- Source & target punctuation symbols ratio
- Source & target proportion of dependency relations between (aligned) constituents
- Source & target difference in depth of their syntactic trees.

In the following image, the architecture of the QUEST++ framework is described.


Pipeline
Raw file
Sentence
Word-Level Feature Extractor
Sentence-Level Feature Extractor
Paragraph
Paragraph-Level Feature Extractor
Sgml file
Document
Document-Level Feature Extractor
Basic classes
Feature Processing Module
Java interface
$f_{11}, f_{12}, \ldots, f_{1m}$
$f_{n1}, f_{n2}, \ldots, f_{nm}$
m - number of features
n - number of t, number of **t** or number of T
Regressors, Classifiers and Feature Selection (sentence- or document-level)
(Python interface)
CRF (word-level)
(C++ interface)
Machine learning interface
Sentence-level model
Document-level model
Word-level model
Pipeline

Picture 2: QUEST++ framework architecture

2.6.4 Neural-based QE methods

Neural network based approaches have been recently used successfully to boost QE performance. NQE systems compared to traditional feature-based QE approaches, seems to be better on modeling non-linear associations between the input and target variables, and hence being able to deliver more generalized and language-independent models. Moreover, unlike shallow QE architectures, NQE systems employ neural networks for processing source and target texts in an end-to-end way.

The idea of explicitly defining QE features as the input for the neural system is not mandatory. Inspired by the encoder-decoder (Sutskever et al., 2014) architecture on machine translation, NQE systems provide a number of encoders to transform a source sentence into a context vector that can later be used on quality score prediction.

NQE architectures can easily be distinguished on one-phase and two-phase systems.

- **One-phase Neural QE systems**

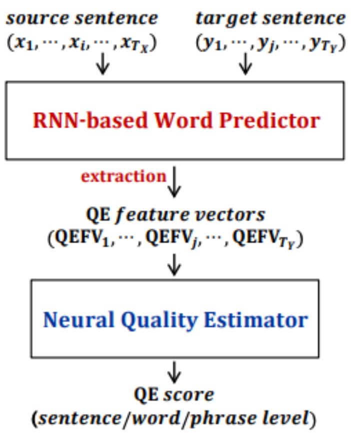
One-phase systems follow a unified architecture with no intermediate stages, trained to produce QE scores in an end-to-end fashion.

- **Two-phase Neural QE systems**

It typically consists of two independently trained neural networks, with the first focusing on extracting features using the source and target sentences as input and the second network aiming at using previously generated features to produce QE scores.

POSTECH Predictor-Estimator (Kim et al., 2017)

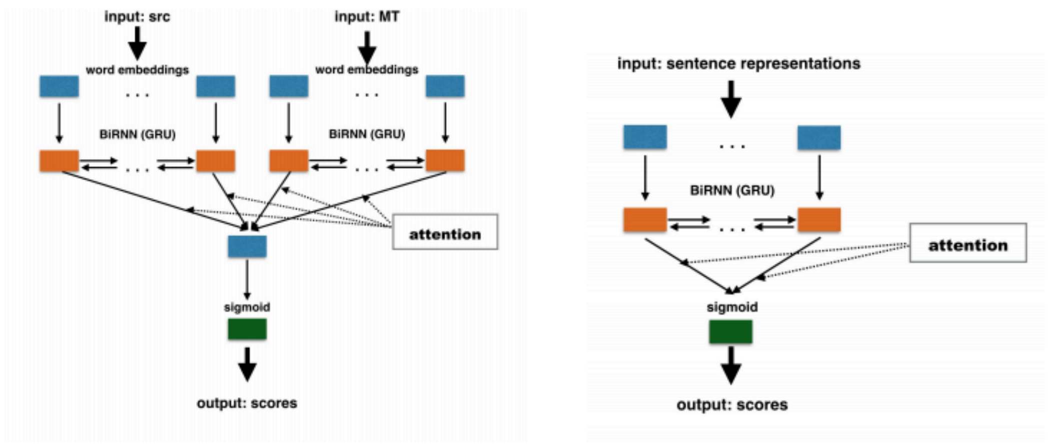
POSTECH is a two-phase purely neural architecture with the best performance at the WMT17 QE shared task, at the levels of word, phrase and sentence prediction. The first part is a bidirectional RNN language model, inspired by the encoder-decoder, known as word predictor. The predictor is trained separately as a pre-task on an additional large-scale parallel corpus to generate QE feature vectors (QEFVs). The second part, known as the Estimator, is a bidirectional RNN that uses QE feature vectors as inputs to predict a multi-level quality estimation score. Note that, to be effective this architecture has to be pre-trained on a very large parallel bilingual corpus that leads to high training requirements in terms of time and processing.


source sentence
$(x_1, \cdots, x_i, \cdots, x_{T_X})$
target sentence
$(y_1, \cdots, y_j, \cdots, y_{T_Y})$
RNN-based Word Predictor
extraction
QE feature vectors
$(\mathrm{QEFV}_1, \cdots, \mathrm{QEFV}_j, \cdots, \mathrm{QEFV}_{T_Y})$
Neural Quality Estimator
QE score
(sentence/word/phrase level)

Picture 3: Predictor-Estimator architecture

deepQuest (Ive et al., 2018)

It is a Neural-based framework that supports all three levels of Quality Estimation and it was developed at the University of Sheffield. It reimplements, the state of the art POS-TECH neural-based architecture to date for sentence-level quality prediction, and BiRNN, a light-weight neural architecture. BiRNN is a one-phase architecture that uses two bidirectional RNNs with Gated Recurrent Units encoders to learn the source-target pair representations. These representations are then being weighted by an attention mechanism generating a vector representation that is afterward used to predict a quality score. One-phase systems, like BiRNN, require only source, MT output, and the golden-standard. Its sentence-level approach can score state-of-the-art results, whereas its document-level approach scores significantly better over previous work.



Picture 4: BiRNN architecture

OpenKiwi (Kepler et al., 2019)

OpenKiwi is an open-source framework developed by Unbabel and is focusing on the task of machine translation quality estimation. It embeds the best-performing systems of the WMT15–18 QE tasks. It supports the training and evaluation of both sentence and word levels QE systems. OpenKiwi reaches near state-of-the-art performance when applied on the En-De SMT and NMT datasets of WMT18 at word and sentence-level tasks. As it is highly customizable, it allows users to combine and modify all systems' key components, while experimenting under the same framework. OpenKiwi addresses the sentence-level QE by implementing the two-following neural-based systems:

- Predictor-Estimator (Kim et al., 2017)
- A stacked ensemble with a linear system (Martins et al., 2016, 2017)

It also includes pre-trained QE models on data from the WMT 2018 campaign, ready for evaluating MT systems. OpeKiwi uses PyTorch as the deep learning framework and it can be imported as a python package in other projects or run through cmd.

3 Data and Methods

In the following chapter, every preprocessing step that took place in the raw machine translation dataset is described in detail, and the feature engineering procedure is extensively explained. More specifically,

In section 3.1 the acquired raw dataset used for the experiments is described.

In section 3.2 the required data-preprocessing steps that took place during the dataset preparation are analyzed.

In section 3.3 the scoring-standard generation for each data point using the TERcom tool is examined.

In section 3.4 the feature-engineering process which includes the training of language models, extraction features and reduction of dimensionality is thoroughly investigated.

3.1 Datasets

3.1.1 TAUS raw dataset

The MT Quality Estimation model was trained on a dataset obtained from the TAUS data repository. TAUS in the role of language data network develops communities and provides data services to buyers and LSPs (language service providers) through a number of applications and APIs including the Matching Data, the Data Cloud and more.

For the purpose of this thesis, a dataset of 77733 raw human-annotated from English to Greek translation segments was extracted from the TAUS data repository. This dataset consists of entries from a variety of different MT engines, industries and content-types which helps on the generalization of the trained model. Despite the large number of columns that the dataset originally had, after the redundant ones were removed and only the necessary ones were kept, the dataset ended up with the following features for each entry:

- the source sentence
- the target sentence
- the human-edited sentence
- the content type of the translation (legal, user manual, marketing material, etc.)
- the industry that it refers to (automotive, healthcare/medical, finance, technology)

- the name of the MT engine from which it was initialized (Microsoft Translator Hub, Google translate)
- the origin, which indicates whether the system it was initialized from is a Machine Translation engine (MT) or a Translation system (TM)
- the match rate, that refers to the percentage by which a translation was suggested by a TM. In case of MT its value is null

3.2 Data Preprocessing

3.2.1 Noise Removal

The initial raw dataset acquired by TAUS consisted of 77720 rows. After further analysis, rows found with empty values in the source, target or in the reference text, considered as invalid were removed. Furthermore, rows with inaccessible or missing values that could not be used in our experiment, were considered as noise and therefore were removed. Excluding noisy rows from the dataset, the number of rows reduced to 77614.

3.2.2 Tokenization

Tokenization is a method in which a given text, document, phrase or sentence is split into smaller units called tokens. These tokens are usually words or numbers or punctuation marks. It is a very common process used for manipulating text in NLP applications such as sentimental analysis, text classification and machine translation. It can be used to transform large texts to numeric vector representations that are more suitable for machine learning and it is also a necessary step before stemming and lemmatization. For the needs of the current implementation, the word tokenizer module from NLTK[6] library and the SpaCy[7] library was tested. After using both libraries into the raw dataset, it was found that the NLTK library performs approximately 37 times faster than Spacy for word tokenization and therefore it was selected.

NLTK word tokenizer	71.44 seconds
Spacy tokenizer	2621 seconds

Time spent on tokenization

[6] https://www.nltk.org/api/nltk.tokenize.html
[7] https://spacy.io/api/tokenizer

3.2.3 Identify translation duplicates using Bifixer

Before any further analysis is applied on our dataset, it had to be ensured that it includes only unique triplets of source, target, reference so that the model will not process the same data multiple times. For that, the Bifixer tool was used. It was developed within the framework of the ParaCrawl (Espì A-Gomis et al., n.d.), EU-funded, project. Bifixer is used to handle duplicated or near-duplicated parallel sentences by appending to each parallel sentence two new fields. A hash, that is produced using the XXHash[8] algorithm, and a ranking score. In case of entries that are identical and share the same hash, the ranking number is used to help the algorithm to choose the best sentence from those.

3.2.4 Remove noisy translation pairs using Bicleaner

Bicleaner is another tool developed in the context of the ParaCrawl project. It is written in Python and it aims at detecting noisy sentence pairs and misalignments in a parallel corpus. It assigns scores to translation pairs, from 0 to 1, which indicates the likelihood of them to be mutual translations. Translation pairs that seem to be very clean are assigned with a score close to 1, whereas sentence pairs that are considered noisy are assigned with a score close to 0. To integrate Bicleaner classifier into the proposed pipeline implementation, all the required steps described here[9] took place. After applying the Bifixer and Bicleaner tools to further clean the dataset, the number of data points was again reduced to 34850.

3.3 Scoring Standard Generation

3.3.1 HTER (Human-targeted Translation Edit Rate)

For the purpose of this research, the HTER (human-targeted translation edit rate) metric was selected as the golden standard. It is the most widely used human-targeted metric for machine translation-related tasks. It has been shown that HTER yields a higher correlation to human judgment than BLEU (Matthew Snover et al., n.d.) at a lower computational cost. It is also used in the WMT[10] (Workshop in Statistical Machine Translation) annual Evaluation Campaigns, as the primary prediction label in the evaluation of quality

[8] http://cyan4973.github.io/xxHash/

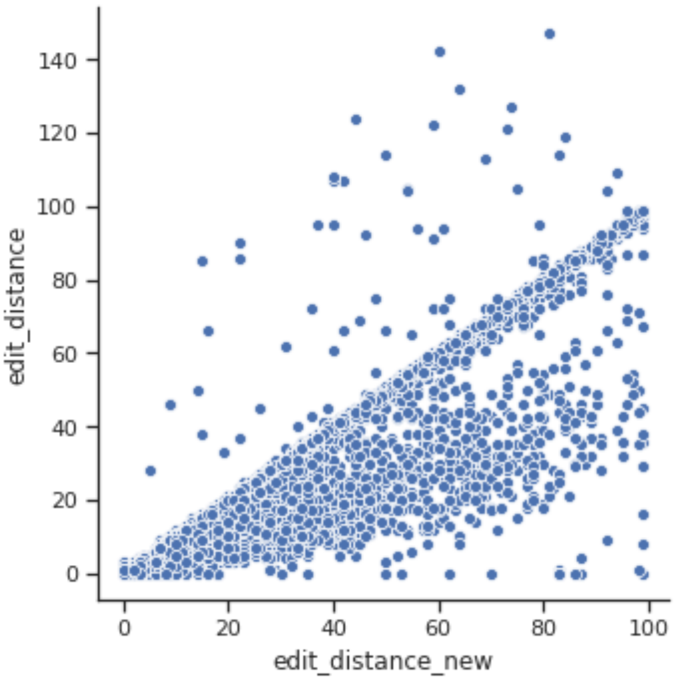
[9] https://github.com/bitextor/bicleaner/#installation--requirements

[10] http://statmt.org/wmt19

estimation task. In order to obtain the number of required edits for every target-reference pair, the TERcom tool was integrated into the pipeline, according to the publisher's documentation instructions, and was applied on the machine-translated and post-edited sentences. Generating scores for the dataset took approximately 80 seconds.

3.3.2 Levenshtein Distance Selection

The Levenshtein distance or edit distance was invented in 1965 by the Russian Mathematician Vladimir Levenshtein (1935-2017) and is used to measure how similar two strings are. It does that, by calculating the required number of single-character substitutions, insertions, or deletions that are required to convert a string into another. Unlike HTER that calculates similarity at a word level, Levenshtein distance does that at a character level. There is a number of cases where QE solutions were developed using the Levenshtein distance as the scoring standard. In our case, we generated and used it to observe the correlation between ED and HTER variables. In the initial raw dataset that was given by TAUS, edit-distance is one of the features for each entry and is calculated by the system in which this particular translation took place, questioning the reliability of its values. To address that, the Levenshtein distance between every translation of the dataset and its corresponding reference was calculated again using a common method. The relationship between the vector of already defined distance values and the vector of newly generated edit distance values can be seen as below.



Picture 5: correlation between edit_distance and edit_distance_new variables

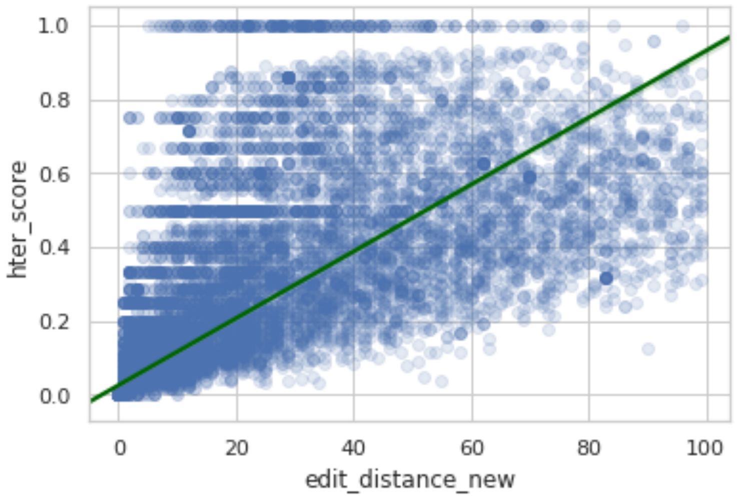
Their high correlation can be confirmed by the values of their Spearman and Pearson correlation coefficient:

```
df[['edit_distance', 'edit_distance_new']].corr(method ='pearson')
                   edit_distance  edit_distance_new
edit_distance           1.000000           0.950822
edit_distance_new       0.950822           1.000000

Pearson's correlation coefficient = 0.950822
Spearman's correlation coefficient = 0.970357
```

Another encouraging observation for using the newly generated edit distance-vector, compared to the given one in the initial dataset, is the higher Pearson's correlation coefficient it has with the HTER scoring-standard vector as seen below:

```
df_clean_data[['edit_distance', 'hter_score']].corr(method ='pearson')
               edit_distance  hter_score
edit_distance       1.000000    0.767724
hter_score          0.767724    1.000000


df_clean_data[['edit_distance_new', 'hter_score']].corr(method ='pearson')
                   edit_distance_new  hter_score
edit_distance_new           1.000000    0.790007
hter_score                  0.790007    1.000000


'edit_distance', 'hter_score' Pearson's correlation coefficient = 0.767724
'edit_distance_new', 'hter_score' Pearson's correlation coefficient = 0.790007
```

The way of how the values of HTER and edit-distance are related can be seen in the following plot:



Picture 6: correlation between edit_distance_new and the target variable hter_score

And the following Pearson's and Spearman's correlation coefficients:

```
Pearson's correlation coefficient = 0.790007
Spearman's correlation coefficient = 0.984032
```

3.4 Feature Engineering

After completing the preprocessing phase, having a clean raw dataset, free of noise data and missing values, the next step is to transform these data into a more digestible format that machine learning algorithms can easier interpret.

3.4.1 Feature Extraction

The process of feature extraction in this work is mainly based on lexical features such as punctuation counts, number of words and word sequences in common and language model probabilities. The feature selection strategy that the current implementation follows, is an extension of another research[11] on QE at sentence level that was made by the students of UVA. In more detail, the following list describes the 61 features that were selected:

Punctuation features

- commaDif - difference in commas
- exclamationDif - difference in exclamation marks
- questionmarkDif - difference in question marks
- dotDif - difference in dots
- hyphenDif - difference in hyphens
- underscoreDif - difference in underscores
- slashDif - difference in slashes
- colonDif - difference in colons
- semicolonDif - difference in semicolons
- capitalCountDif - difference in capital letters
- misMatch - mismatches in brackets or accolades
- wordCountSrc - number of words in the source sentence
- wordCountTgt - number of words in the target sentence
- wordCountDif - difference in the number of tokens

Statistical features

- commaDifNorm - difference in commas, normalized
- exclamationDifNorm - difference in exclamation, normalized by the amount of characters in target sentence

[11] https://github.com/LucSkyvvalker/TAUS

- questionmarkDifNorm - difference in question marks, normalized by the amount of characters in target sentence
- dotDifNorm - difference in dots, normalized by the amount of characters in target sentence
- hyphenDifNorm - difference in hyphens, normalized by the amount of characters in target sentence
- underscoreDifNorm - difference in underscores, normalized by the amount of characters in target sentence
- slashDifNorm - difference in slashes, normalized by the amount of characters in target sentence
- colonDifNorm - difference in colons, normalized by the amount of characters in target sentence
- semicolonDifNorm - difference in semicolons, normalized by the amount of characters in target sentence

Lexical features

- verbDif - difference in verbs using spaCy POS tagger
- nounDif - difference in nouns using spaCy POS tagger

Language model features

- logPerpSrc0 - source sentence unigram LM perplexity, trained on the europarl v7 large corpus of the source language, transformed with base 2 log
- logPerpTgt0 - target sentence unigram LM perplexity, trained on the europarl v7 large corpus of the target language, transformed with base 2 log
- logPerpSrc1 - source sentence bigram LM perplexity, trained on the europarl v7 large corpus of the source language, transformed with base 2 log
- logPerpTgt1 - target sentence bigram LM perplexity, trained on the europarl v7 large corpus of the target language, transformed with base 2 log
- logPerpSrc2 - source sentence trigram LM perplexity, trained on the europarl v7 large corpus of the source language, transformed with base 2 log
- logPerpTgt2 - target sentence trigram LM perplexity, trained on the europarl v7 large corpus of the target language, transformed with base 2 log
- logPerpSrcPos0 - source sentence unigram LM perplexity, trained on the POS tagged version of the europarl v7 large corpus of the source language, transformed with base 2 log

- logPerpTgtPos0 - target sentence unigram LM perplexity, trained on the POS tagged version of the europarl v7 large corpus of the target language, transformed with base 2 log
- logPerpSrcPos1 - source sentence bigram LM perplexity, trained on the POS tagged version of the europarl v7 large corpus of the source language, transformed with base 2 log
- logPerpTgtPos1 - target sentence bigram LM perplexity, trained on the POS tagged version of the europarl v7 large corpus of the target language, transformed with base 2 log
- logPerpSrcPos2 - source sentence trigram LM perplexity, trained on the POS tagged version of the europarl v7 large corpus of the source language, transformed with base 2 log
- logPerpTgtPos2 - target sentence trigram LM perplexity, trained on the POS tagged version of the europarl v7 large corpus of the target language, transformed with base 2 log
- logProbSrc0 - source sentence unigram LM probability, trained on the europarl v7 large corpus of the source language, transformed with base 2 log
- logProbTgt0 - target sentence unigram LM probability, trained on the europarl v7 large corpus of the target language, transformed with base 2 log
- logProbSrc1 - source sentence bigram LM probability, trained on the europarl v7 large corpus of the source language, transformed with base 2 log
- logProbTgt1 - target sentence bigram LM probability, trained on the europarl v7 large corpus of the target language, transformed with base 2 log
- logProbSrc2 - source sentence trigram LM probability, trained on the europarl v7 large corpus of the source language, transformed with base 2 log
- logProbTgt2 - target sentence trigram LM probability, trained on the europarl v7 large corpus of the target language, transformed with base 2 log
- logProbSrcPos0 - source sentence unigram LM probability, trained on the euro-parl v7 large corpus of the source language, transformed with base 2 log
- logProbTgtPos0 - target sentence unigram LM probability, trained on the europarl v7 large corpus of the target language, transformed with base 2 log
- logProbSrcPos1 - source sentence bigram LM probability, trained on the europarl v7 large corpus of the source language, transformed with base 2 log

- logProbTgtPos1 - target sentence bigram LM probability, trained on the europarl v7 large corpus of the target language, transformed with base 2 log
- logProbSrcPos2 - source sentence trigram LM probability, trained on the europarl v7 large corpus of the source language, transformed with base 2 log
- logProbTgtPos2 - target sentence trigram LM probability, trained on the europarl v7 large corpus of the target language, transformed with base 2 log
- unk_ngramsSrc0 - unigram LM probability for unknown words in the source sentence, trained on the europarl v7 large corpus of the source language, transformed with base 2 log
- unk_ngramsTgt0 - unigram LM probability for unknown words in the target sentence, trained on the europarl v7 large corpus of the target language, transformed with base 2 log
- unk_ngramsSrc1 - bigram LM probability for unknown words in the source sentence, trained on the europarl v7 large corpus of the source language, transformed with base 2 log
- unk_ngramsTgt1 - bigram LM probability for unknown words in the target sentence, trained on the europarl v7 large corpus of the target language, transformed with base 2 log
- unk_ngramsSrc2 - trigram LM probability for unknown words in the source sentence, trained on the europarl v7 large corpus of the source language, transformed with base 2 log
- unk_ngramsTgt2 - trigram LM probability for unknown words in the target sentence, trained on the europarl v7 large corpus of the target language, transformed with base 2 log
- unk_ngramsSrcPos0 - unigram LM probability for unknown words in the POS taged source sentence, trained on the POS tagged version of the europarl v7 large corpus of the source language, transformed with base 2 log
- unk_ngramsTgtPos0 - unigram LM probability for unknown words in the POS taged target sentence, trained on the POS tagged version of the europarl v7 large corpus of the target language, transformed with base 2 log
- unk_ngramsSrcPos1 - bigram LM probability for unknown words in the POS taged source sentence, trained on the POS tagged version of the europarl v7 large corpus of the target language, transformed with base 2 log

- unk_ngramsTgtPos1 - bigram LM probability for unknown words in the POS taged target sentence, trained on the POS tagged version of the europarl v7 large corpus of the target language, transformed with base 2 log
- unk_ngramsSrcPos2 - trigram LM probability for unknown words in the POS taged source sentence, trained on the POS tagged version of the europarl v7 large corpus of the source language, transformed with base 2 log
- unk_ngramsTgtPos2 - trigram LM probability for unknown words in the POS taged target sentence, trained on the POS tagged version of the europarl v7 large corpus of the target language, transformed with base 2 log

3.4.2 Feature Generation

Language Models Training

In contrast to the quite straight-forward way that some of the listed above features were generated (punctuation, lexical features), to generate the n-gram language model features a more demanding process took place. In more detail, a unigram, a bigram and a trigram LMs were trained on both the source and target language texts and their POS-tagged versions, ending up with a total of 12 different n-gram language models. The bigram and trigram language models were built using the KenLM (Heafield, 2011) LM implementation, whereas for the unigram language models, an implementation provided by previously mentioned project[10] was used. For the training of these LMs, the Europarl v7 Greek-English parallel corpus was selected. More specifically, this large bilingual corpus was divided into two monolingual corpora and models that were addressed for source-language, were trained on the English corpus, whereas models related to the target-language were trained on the Greek monolingual corpus.

As data sparsity is almost always an issue in statistical modeling, LMs were required to adopt a smoothing method to improve their prediction performance and avoid zero probabilities by unseen n-grams. To do so, the method of modified Kneser-Ney smoothing (Heafield et al., 2013) was applied.

A detailed analysis on how n-gram LMs function and the required resources for their training follows below.

KenLM Language Models

KenLM is a fast and low-memory language modeling toolkit. It implements two data structures for fast LM queries, achieving substantial reductions in time and memory cost. The "PROBING" data-structure is designed for speed, being 2,4 times faster than the widely known SRILM only by using the 57% of the memory. On the other hand, "TRIE" data structure is designed for low memory consumption. It is open-source and is offered on both C++ and Java interfaces.

European Parliament Parallel Corpus v7

For the training of the language models (LMs), which are described in more detail in a later section, the Greek–English Europarl parallel corpus v7 (Koehn, n.d.) was used. Europarl is a parallel corpus extracted from the European Parliament proceedings and is available for 21 European languages including Romanic, Germanic, Slavic, Finni-Ugric, Baltic languages and Greek. The Greek–English parallel corpus consists of 1,235,976 bilingual sentence pairs. Before feeding the corpora to LMs, some preparation was required. At first, these texts were tokenized using NLTK's word tokenizer. And then, the PoS-tagged corpora was generated out of the original texts, using the "el_core_news_md" and the "en_core_web_lg" pre-trained statistical models of spaCy[12] library.

Table 1: Greek-English Europarl v7 parallel corpus counts

	English	Greek
Sentence count	1,235,976	1,235,976
Tokens count	31,953,210	37,122,787
Vocabulary	123,521	260,521

N-gram Language Models

The N-gram model is the simplest form of language models and its role is to assign probabilities to sentences and sequences of words. The intuition behind n-gram LMs is to compute the probability of a word to occur given all the previous n words, by using only the conditional probability of n previous words. In that way, it approximates the history of a word given only the last few terms, instead of computing the probability using its entire history. In case of bigrams for instance, the probability of a term to occur is predicted using only the conditional probability of its previous word.

$$P(w_n|w_{n-1})$$

This approximation is also known as the Markov assumption.

$$P(w_n|w_1^{n-1}) \approx P(w_n|w_{n-1})$$

By generalizing the bigram model, the trigram and therefore the n-gram model can be easily produced. The equation that expresses the general n-gram approximation for the conditional probability of the next word in a sequence of words is

$$P(w_n|w_1^{n-1}) \approx P(w_n|w_{n-N+1}^{n-1})$$

The decomposition of the probability of entire sequences, using the chain rule of probability combined with the previously mentioned bigram assumption, allows us to compute the probability of a complete sequence of words.

[12] https://spacy.io/usage/models

$$P(w_1^n) \approx \prod_{k=1}^{n} P(w_k|w_{k-1})$$

For the estimation of each n-gram probability, the maximum likelihood estimation or MLE method is used. As such, in order to calculate a bigram probability of a word w_{n-1} followed by a word w_n, we divide the count of the bigram $C(w_{n-1},w_n)$ with the sum of all the bigrams that have the same first word w_{n-1}.

$$P(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n)}{C(w_{n-1})}$$

To avoid numerical underflow while multiplying the number of n-grams together (since probabilities range from 0 to 1) the language model probabilities are converted to log probabilities.

Smoothing

Like many statistical models, n-grams are significantly dependent on the training corpus. As a fact, MLE based models are exposed to the issue of data sparsity. More specifically, any n-grams in a querying sentence that don't appear in the corpus during the training process are assigned with a 0 probability. This is happening as LMs cannot cover all the possible n-grams that could appear in a language no matter how large the corpus they were trained on is. However, this is incorrect as perfectly acceptable word sequences can be, incorrectly, considered by the model as impossible events. To face that, the method of smoothing is introduced. Smoothing aims at moving a bit of probability from more frequent events to unseen events. Even though there are multiple techniques (Laplace, Add-k smoothing, etc.) to apply smoothing, the current solution implements the modified Kneser-Ney (Heafield et al., 2013) smoothing.

LM Evaluation

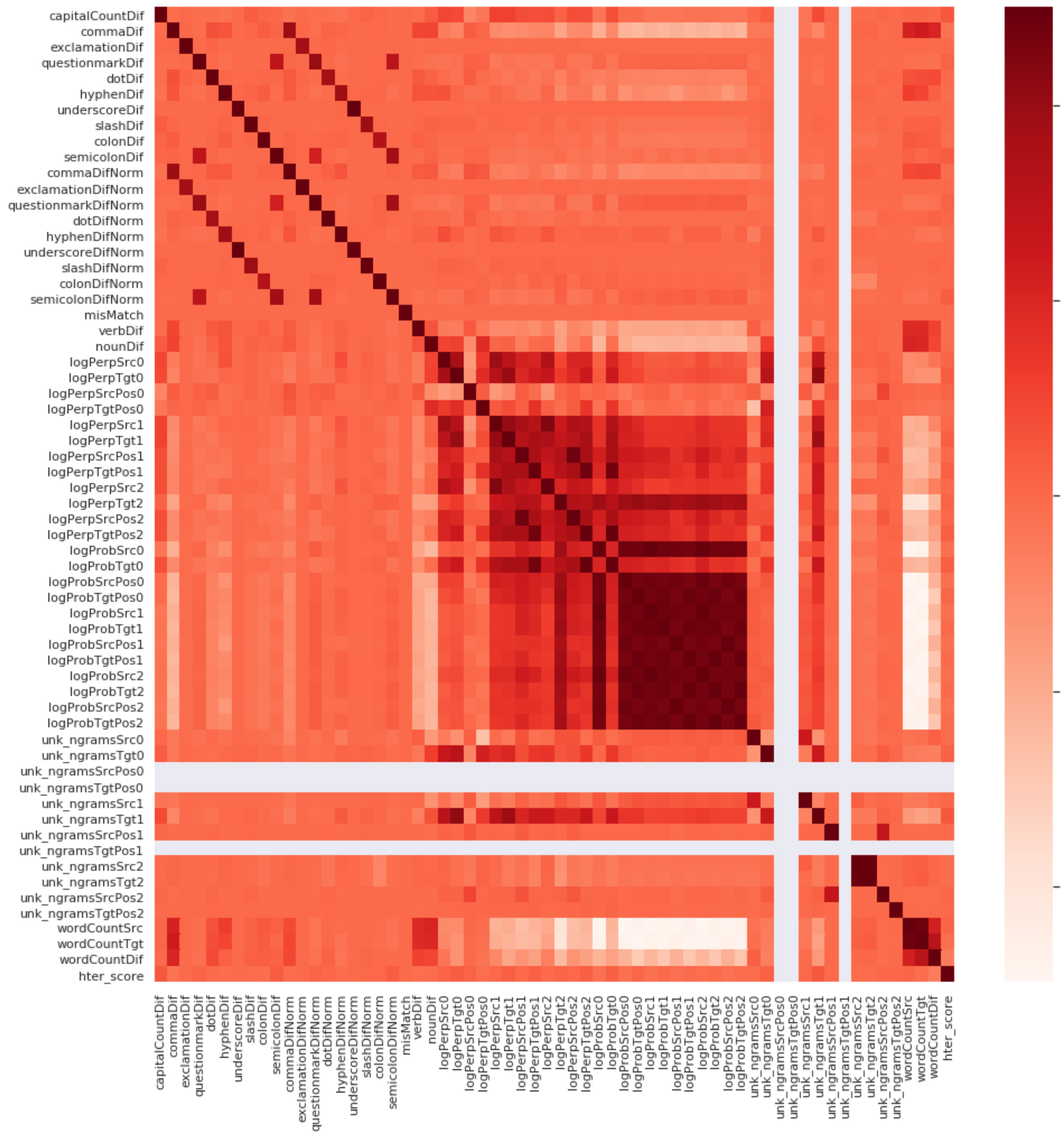
Perplexity is a metric used for the evaluation of language models. In general, as a measurement, it indicates how well a model predicts a test set. The perplexity for a test set W = $w_1,w_2 \ldots w_N$ is the multiplicative inverse probability that is given by a model to the test set, normalized by the number of words in it.

$$\mathrm{PP}(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i|w_1 \ldots w_{i-1})}}$$

From the equation above, it is clear that the higher the conditional probability of a sequence of words, the lower the perplexity is.

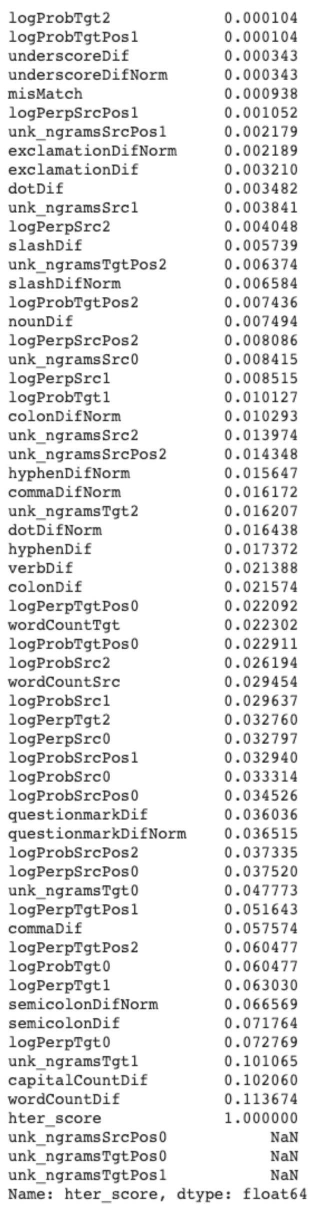
3.4.3 Feature Evaluation

Before moving forward into the machine learning regression process, an evaluation analysis was performed on the 61 extracted features. In order to identify linear relationships between the selected features, the following correlation matrix, based on Pearson's correlation coefficient, was generated.



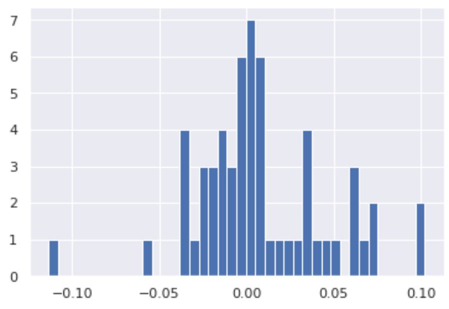
Picture 7: correlation matrix heatmap showing the correlation between all the features

Furthermore, a correlation attribute evaluation was implemented in order to identify which features reveal a higher absolute correlation with the "hter_score" target variable. The results are presented in the image below.



```
logProbTgt2              0.000104
logProbTgtPos1           0.000104
underscoreDif            0.000343
underscoreDifNorm        0.000343
misMatch                 0.000938
logPerpSrcPos1           0.001052
unk_ngramsSrcPos1        0.002179
exclamationDifNorm       0.002189
exclamationDif           0.003210
dotDif                   0.003482
unk_ngramsSrc1           0.003841
logPerpSrc2              0.004048
slashDif                 0.005739
unk_ngramsTgtPos2        0.006374
slashDifNorm             0.006584
logProbTgtPos2           0.007436
nounDif                  0.007494
logPerpSrcPos2           0.008086
unk_ngramsSrc0           0.008415
logPerpSrc1              0.008515
logProbTgt1              0.010127
colonDifNorm             0.010293
unk_ngramsSrc2           0.013974
unk_ngramsSrcPos2        0.014348
hyphenDifNorm            0.015647
commaDifNorm             0.016172
unk_ngramsTgt2           0.016207
dotDifNorm               0.016438
hyphenDif                0.017372
verbDif                  0.021388
colonDif                 0.021574
logPerpTgtPos0           0.022092
wordCountTgt             0.022302
logProbTgtPos0           0.022911
logProbSrc2              0.026194
wordCountSrc             0.029454
logProbSrc1              0.029637
logPerpTgt2              0.032760
logPerpSrc0              0.032797
logProbSrcPos1           0.032940
logProbSrc0              0.033314
logProbSrcPos0           0.034526
questionmarkDif          0.036036
questionmarkDifNorm      0.036515
logProbSrcPos2           0.037335
logPerpSrcPos0           0.037520
unk_ngramsTgt0           0.047773
logPerpTgtPos1           0.051643
commaDif                 0.057574
logPerpTgtPos2           0.060477
logProbTgt0              0.060477
logPerpTgt1              0.063030
semicolonDifNorm         0.066569
semicolonDif             0.071764
logPerpTgt0              0.072769
unk_ngramsTgt1           0.101065
capitalCountDif          0.102060
wordCountDif             0.113674
hter_score               1.000000
unk_ngramsSrcPos0             NaN
unk_ngramsTgtPos0             NaN
unk_ngramsTgtPos1             NaN
Name: hter_score, dtype: float64
```

Picture 8: correlation between all features and the target variable hter_score



Picture 9: frequency distribution of correlation coefficients

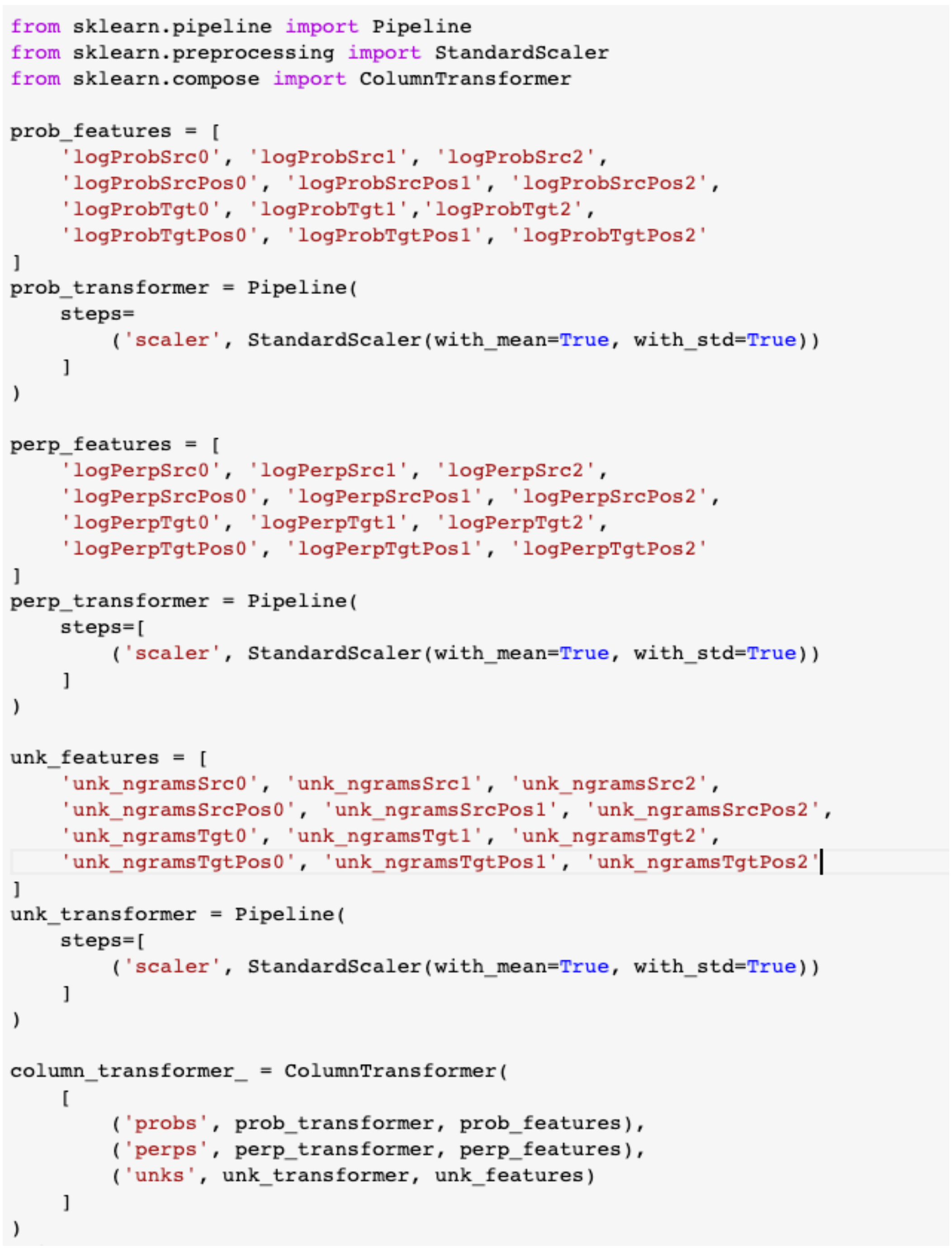
By observing the plots above, it is obvious that there are many features whose correlation coefficient is really low, and thus could be removed. Further analysis is performed in the following section examining the way to reduce the dimensionality of the dataset.

3.4.4 Dimensionality Reduction

High dimensionality in Machine Learning can often cause multiple issues. The reason that these issues occur is that when the dimensionality of a dataset increases, the volume of the space increases so fast that the available data becomes sparse, and therefore models are led to high computing time and overfitting. There are multiple different approaches to apply on a dataset in order to reduce the number of its features. In the current work, the method of Principal Component Analysis (PCA) was chosen.

Principle Component Analysis

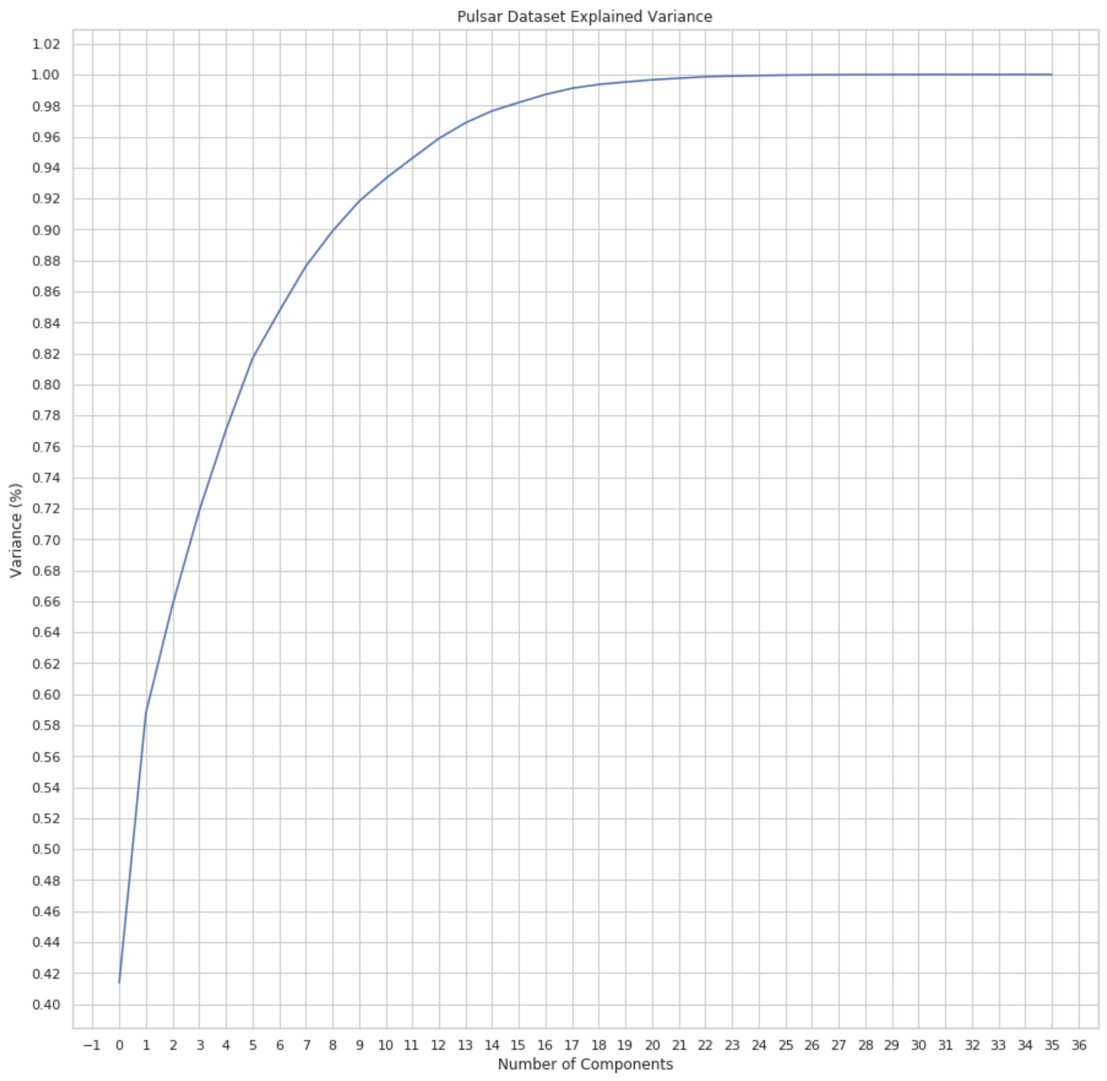
The idea of Principal Component Analysis is to transform a dataset of possibly correlated variables, into a dataset of linearly uncorrelated variables called principal components, while keeping the variance of the transformed data set as high as possible. A requirement for applying PCA is to define the number of principal components. To find out this number, several steps had to take place. First, we transformed our data using the StandardScaler and the ColumnTransformer to apply standard scaling on groups of columns. Below the piece of code responsible for the scaling process is presented.



```python
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.compose import ColumnTransformer

prob_features = [
    'logProbSrc0', 'logProbSrc1', 'logProbSrc2',
    'logProbSrcPos0', 'logProbSrcPos1', 'logProbSrcPos2',
    'logProbTgt0', 'logProbTgt1','logProbTgt2',
    'logProbTgtPos0', 'logProbTgtPos1', 'logProbTgtPos2'
]
prob_transformer = Pipeline(
    steps=
        ('scaler', StandardScaler(with_mean=True, with_std=True))
    ]
)

perp_features = [
    'logPerpSrc0', 'logPerpSrc1', 'logPerpSrc2',
    'logPerpSrcPos0', 'logPerpSrcPos1', 'logPerpSrcPos2',
    'logPerpTgt0', 'logPerpTgt1', 'logPerpTgt2',
    'logPerpTgtPos0', 'logPerpTgtPos1', 'logPerpTgtPos2'
]
perp_transformer = Pipeline(
    steps=[
        ('scaler', StandardScaler(with_mean=True, with_std=True))
    ]
)

unk_features = [
    'unk_ngramsSrc0', 'unk_ngramsSrc1', 'unk_ngramsSrc2',
    'unk_ngramsSrcPos0', 'unk_ngramsSrcPos1', 'unk_ngramsSrcPos2',
    'unk_ngramsTgt0', 'unk_ngramsTgt1', 'unk_ngramsTgt2',
    'unk_ngramsTgtPos0', 'unk_ngramsTgtPos1', 'unk_ngramsTgtPos2'
]
unk_transformer = Pipeline(
    steps=[
        ('scaler', StandardScaler(with_mean=True, with_std=True))
    ]
)

column_transformer_ = ColumnTransformer(
    [
        ('probs', prob_transformer, prob_features),
        ('perps', perp_transformer, perp_features),
        ('unks', unk_transformer, unk_features)
    ]
)
```

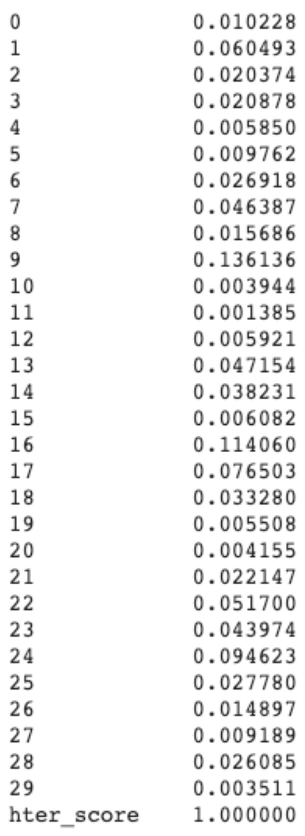
Picture 10: Scaling applied on groups of features

After having the dataset scaled, the next step was to measure the explained variance of the existing features using the sklearn library and plot it using the matplotlib library. The results can be seen below.



Picture 10: Relationship between the number of components and the amount of variance they cover

In the plot above, it can be observed that approximately the 100% of the total variance of the data could be explained by using 30 components. After implementing the PCA using the best 30 components, we generated again the correlation attribute evaluation to illustrate the correlation of the final components with the "hter_score" target variable. The results are presented below.



```
0             0.010228
1             0.060493
2             0.020374
3             0.020878
4             0.005850
5             0.009762
6             0.026918
7             0.046387
8             0.015686
9             0.136136
10            0.003944
11            0.001385
12            0.005921
13            0.047154
14            0.038231
15            0.006082
16            0.114060
17            0.076503
18            0.033280
19            0.005508
20            0.004155
21            0.022147
22            0.051700
23            0.043974
24            0.094623
25            0.027780
26            0.014897
27            0.009189
28            0.026085
29            0.003511
hter_score    1.000000
```

Picture 11: correlation between the new components/features and the target variable hter_score

4 Experiments

The main scope of this chapter was to explore various regression models, trained to estimate the quality of translations from English to Greek. In order to find the most useful and effective algorithm, many trials on various models were performed.
Before applying any machine learning, our dataset was split randomly into the train set and the test set. For the train set the 80% of the initial dataset was used, whereas for the test set, the rest 20% was kept.

4.1 Hyperparameter optimization

As the performance of ML models depends heavily on the hyperparameters, an important task was to manage to select optimal values during the training process. For that, the process of hyper-parameter tuning took place using the GridSearcCV that is provided by the sklearn library with 5-fold cross-validation. Grid search CV trains a machine learning model by testing multiple combinations of training hyperparameters over a cross-validation procedure and then selects the combination of parameters that optimizes the desired evaluation metric. The experiments were conducted using a train set of 27879 entries and 30 features (principal components) on the following 4 regression algorithms.
Support Vector Machine Regressor

- Support Vector Machine Regressor
- Random Forest Regressor
- Multilayer Perceptron Regressor
- Extreme Gradient Boosted Trees Regressor

4.2 Support Vector Machine

Although SVM has been used extensively (Felice & Specia, 2012; Langlois, 2015) in the bibliography with success, in the current experiments it didn't manage to achieve optimum results. More specifically, using the following hyper-parameters:

Table 2: SVM hyperparameter values

kernel	Rbf
C	1000
gama	0.01

SVR achieved the following scores:

Table 3: SVM regression scores

Pearson correlation coefficient r	0.376
Spearman correlation coefficient ρ	0.381
Mean Absolute Error (MAE)	0.136
Root Mean Square Error (RMSE)	0.205

4.3 Random Forest

Random Forests (Tezcan et al., 2016)

Using the following hyper-parameters:

Table 4: RF hyperparameters values

bootstrap	true
max_depth	200
max_features	auto
min_samples_leaf	2
min_samples_split	4
n_estimators	1000

RF achieved the following scores:

Table 5: RF regression scores

Pearson correlation coefficient r	0.715
Spearman correlation coefficient ρ	0.693
Mean Absolute Error (MAE)	0.079
Root Mean Square Error (RMSE)	0.153

4.4 Multilayer Perceptron

Using the following hyper-parameters:

Table 6: MLP hyperparameters values

Solver	adam
hidden_layer_sizes	800
max_iter	1000

MLP achieved the following scores:

Table 7: MLP regression scores

Pearson correlation coefficient r	0.561
Spearman correlation coefficient ρ	0.505
Mean Absolute Error (MAE)	0.121
Root Mean Square Error (RMSE)	0.186

4.5 eXtreme Gradient Boosting

Using the following hyper-parameters:

Table 8: XGBoost hyperparameters values

colsample_bytree	0.8
gamma	0.5
learning_rate	0.02
max_depth	10
n_estimators	500
objective	reg:squarederror
subsample	0.8

XGBoost achieved the following scores:

Table 9: XGBoost regression scores

Pearson correlation coefficient r	0.652
Spearman correlation coefficient ρ	0.623
Mean Absolute Error (MAE)	0.108
Root Mean Square Error (RMSE)	0.169

5 Conclusions

5.1 Conclusions

This work presented an application of machine learning models trying to estimate the quality of machine translation outputs. To do so, several algorithms were used and gave interesting results. The implementation of this project is divided in 3 main phases. The first is the phase of the dataset preprocessing, the second is the phase of the language model training and feature extraction and the last phase is focused on the machine learning experiments. From the scores of the models in Chapter 4, it can be seen that, for the given dataset and language pair, the Random Forest regressor achieved the highest Pearson correlation with the target variable and thus, is considered to perform better than other models on this task. On the other hand, since the features that were used during the model training are based mainly on lexical similarity, a high score does not guarantee a high similarity in the meaning between the source and target sentences.

5.2 Further work

Further improvement in the current work can be achieved in multiple ways. By training Language Models on larger corpora, more accurate n-gram probabilities can be generated and therefore construct better features. Finally, since this solution is not focusing as much as it should into the context of the texts, a way to improve the semantic performance of the model would be to incorporate Neural Networks in it.

2019 Fourth Conference on Machine Translation (WMT19). (n.d.). Retrieved January 14, 2020, from https://www.statmt.org/wmt19/

Albrecht, J. S., & Hwa, R. (2007). A re-examination of machine learning approaches for sentence-level MT evaluation. *ACL 2007 - Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*.

Arnold, D. (2003). *Why translation is difficult for computers*. https://doi.org/10.1075/btl.35.11arn

Avramidis, E. (2013). RankEval: Open Tool for Evaluation of Machine-Learned Ranking. *The Prague Bulletin of Mathematical Linguistics*. https://doi.org/10.2478/pralin-2013-0012

Bach, N., Huang, F., & Al-Onaizan, Y. (2011). Goodness: A method for measuring machine translation confidence. *ACL-HLT 2011 - Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*.

Barrault, L., Bojar, O., Costa-jussà, M. R., Federmann, C., Fishel, M., Graham, Y., Haddow, B., Huck, M., Koehn, P., Malmasi, S., Monz, C., Müller, M., Pal, S., Post, M., & Zampieri, M. (2019). *Findings of the 2019 Conference on Machine Translation (WMT19)*. *2*(1), 1–61. https://doi.org/10.18653/v1/w19-5301

Bawden, R., Bretonnel Cohen, K., Grozea, C., Jimeno Yepes, A., Kittner, M., Krallinger, M., Mah, N., Neveol, A., Neves, M., Soares, F., Siu, A., Verspoor, K., & Vicente Navarro, M. (2019). *Findings of the WMT 2019 Biomedical Translation Shared Task: Evaluation for MEDLINE Abstracts and Biomedical Terminologies*. 29–53. https://doi.org/10.18653/v1/w19-5403

Blatz, J., Fitzgerald, E., Foster, G., Gandrabur, S., Goutte, C., Kulesza, A., Sanchis, A., & Ueffing, N. (2004). *Confidence estimation for machine translation*. 315-es. https://doi.org/10.3115/1220355.1220401

Chatterjee, R., Federmann, C., Negri, M., & Turchi, M. (2019). *Findings of the WMT 2019 Shared Task on Automatic Post-Editing*. 11–28. https://doi.org/10.18653/v1/w19-5402

Doddington, G. (2002). *Automatic evaluation of machine translation quality using n-gram co-occurrence statistics*. https://doi.org/10.3115/1289189.1289273

Espì A-Gomis, M., Forcada, M. L., & Hoang, H. (n.d.). *ParaCrawl: Web-scale parallel corpora for the languages of the EU*. Retrieved January 6, 2020, from https://paracrawl.eu/releases.html

Felice, M., & Specia, L. (2012). Linguistic Features for Quality Estimation. *Proceedings of the Seventh Workshop on Statistical Machine Translation*.

Garg, A., & Agarwal, M. (2018). *Machine Translation: A Literature Review*. http://arxiv.org/abs/1901.01122

Giménez, J., & Màrquez, L. (2008). Heterogeneous automatic MT evaluation through non-parametric metric combinations. *Proceedings of the Third International Joint Conference on Natural Language Processing (IJCNLP)*, *2003*, 319–326. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.142.1512

Han, L. (2016). *Machine Translation Evaluation Resources and Methods: A Survey*. *2018*. http://arxiv.org/abs/1605.04515

He, Y., Ma, Y., Van Genabith, J., & Way, A. (2010). Bridging SMT and TM with translation recommendation. *ACL 2010 - 48th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference*.

Heafield, K. (2011). KenLM : Faster and Smaller Language Model Queries. *Proceedings of the Sixth Workshop on Statistical Machine Translation*.

Heafield, K., Pouzyrevsky, I., Clark, J. H., & Koehn, P. (2013). Scalable modified Kneser-Ney language model estimation. *ACL 2013 - 51st Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference*.

Ive, J., Blain, F., & Specia, L. (2018). deepQuest: A Framework for Neural-based Quality Estimation. *Proceedings of the 27th International Conference on Computational Linguistics*.

Kepler, F., Trénous, J., Treviso, M., Vera, M., & Martins, A. F. T. (2019). *OpenKiwi: An Open Source Framework for Quality Estimation*. https://doi.org/10.18653/v1/p19-3020

Kim, H., Lee, J.-H., & Na, S.-H. (2017). *Predictor-Estimator using Multilevel Task Learning with Stack Propagation for Neural Quality Estimation* (Vol. 2). http://www.statmt.org/wmt17/translation-task.html

Koehn, P. (n.d.). *Europarl: A Parallel Corpus for Statistical Machine Translation*. Retrieved January 3, 2020, from http://www.europarl.eu.int/

Koehn, P., Guzmán, F., Chaudhary, V., & Pino, J. (2019). *Findings of the WMT 2019 Shared Task on Parallel Corpus Filtering for Low-Resource Conditions*. 54–72. https://doi.org/10.18653/v1/w19-5404

Langlois, D. (2015). *LORIA System for the WMT15 Quality Estimation Shared Task*. https://doi.org/10.18653/v1/w15-3038

Lavie, A., & Agarwal, A. (2007). METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgments. *Proceedings of the Second Workshop on Statistical Machine Translation*.

Li, X., Michel, P., Anastasopoulos, A., Belinkov, Y., Durrani, N., Firat, O., Koehn, P., Neubig, G., Pino, J., & Sajjad, H. (2019). *Findings of the First Shared Task on Machine Translation Robustness*. 91–102. https://doi.org/10.18653/v1/w19-5303

Lin, C.-Y. (2004). Looking for a Few Good Metrics: ROUGE and its Evaluation. *NTCIR Workshop*.

Luong, N. Q., Besacier, L., & Lecouteux, B. (2015). *Word Confidence Estimation for SMT N-best List Re-ranking*. 1–9. https://doi.org/10.3115/v1/w14-0301

Martin, A. F., & Przybocki, M. A. (n.d.). *NIST 2003 Language Recognition Evaluation*.

Martins, A. F. T., Astudillo, R., Hokamp, C., & Kepler, F. (2016). *Unbabel's Participation in the WMT16 Word-Level Translation Quality Estimation Shared Task*. https://doi.org/10.18653/v1/w16-2387

Martins, A. F. T., Junczys-Dowmunt, M., Kepler, F. N., Astudillo, R., Hokamp, C., & Grundkiewicz, R. (2017). Pushing the Limits of Translation Quality Estimation. *Transactions of the Association for Computational Linguistics*, *5*, 205–218. https://doi.org/10.1162/tacl_a_00056

Montgomery, D. C., & Runger, G. C. (1994). Applied Statistics and Probability for Engineers. *European Journal of Engineering Education*. https://doi.org/10.1080/03043799408928333

Papineni, K., Roukos, S., Ward, T., & Zhu, W.-J. (n.d.). *BLEU: a Method for Automatic Evaluation of Machine Translation*.

Pearson, K. (1895). VII. Note on regression and inheritance in the case of two parents. *Proceedings of the Royal Society of London*. https://doi.org/10.1098/rspl.1895.0041

Snover, M., Madnani, N., Dorr, B., & Schwartz, R. (2008). TERp system description. *MetricsMATR Workshop at AMTA*.

Snover, Matthew, Dorr, B., Schwartz, R., Micciulla, L., & Makhoul, J. (n.d.). *A Study of Translation Edit Rate with Targeted Human Annotation*.

Song, X., & Cohn, T. (2011). Regression and ranking based optimisation for sentence level machine translation evaluation. *... the Sixth Workshop on Statistical Machine Translation*, 123–129. http://dl.acm.org/citation.cfm?id=2132975

Soricut, R., & Echihabi, A. (2010). TrustRank: Inducing trust in automatic translations via ranking. *ACL 2010 - 48th Annual Meeting of the Association for Computational*

Linguistics, Proceedings of the Conference.

Specia, L, Turchi, M., … N. C.-13th C. of the, & 2009, undefined. (n.d.). Estimating the sentence-level quality of machine translation systems. *Academia.Edu*. Retrieved January 3, 2020, from http://www.academia.edu/download/30664504/eamt.pdf#page=44

Specia, Lucia, Paetzold, G. H., & Scarton, C. (2015). Multi-level translation quality prediction with QUEST++. *ACL-IJCNLP 2015 - 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing, Proceedings of System Demonstrations*. https://doi.org/10.3115/v1/p15-4020

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. *Advances in Neural Information Processing Systems*.

Tezcan, A., Hoste, V., & Macken, L. (2016). *UGENT-LT3 SCATE Submission for WMT16 Shared Task on Quality Estimation*. https://doi.org/10.18653/v1/w16-2393

Tillmann, C., Vogel, S., Ney, H., Zubiaga, A., & Sawaf, H. (1997). Accelerated {DP}~Based Search for Statistical Translation. *Fifth European Conference\ on Speech Communication and Technology*.

Turian, J. P., Shen, L., & Melamed, I. D. (2003). Evaluation of Machine Translation and its Evaluation. *Proceedings of MT Summit IX*.

Vidal, E. (1997). Finite-state speech-to-speech translation. *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*. https://doi.org/10.1109/icassp.1997.599563