Named Entity Recognition for Model Quality Estimation

Slimane Mesbah

Named Entity Recognition for Model Quality Estimation

Slimane Mesbah, supervised by Francois Yvon

ISIR UPMC AND PARIS SACLAY UNIVERSITY

April 12, 2023

Abstract

This paper proposes a new metric that combines Meteor with a corrected version of Nist to address the problem of Nist's dependence on sentence length. While alternative metrics like Quest++, and Transquest have been developed to evaluate MT systems without relying on reference translations, they are still trained on scores given by reference translations. Therefore, improving symbolic metrics like Bleu, Meteor, or Nist is important to train these state-of-the-art metrics on well-processed data. The article also suggests editing named entities as an approach to improving the performance of Transquest. Code is available at: **https://github.com/slimane-msb/TransQuest/tree/master/NER**

Contents

1 Introduction

The lack of reference translations for Machine Translation (MT) systems has led to the development of alternative metrics, such as Questplusplus, deepQuest, and Transquest, which do not rely on reference translations to evaluate the quality of MT systems. However, these metrics are still trained on scores given by reference translations, which makes them not completely independent of reference translations.

Therefore, it is still important to improve symbolic metrics like Bleu, Meteor, or Nist, which can accurately evaluate translations, in order to train these state-of-the-art metrics such transquest on well-processed data. In this article, we propose a new metric that combines Meteor with a corrected version of Nist to address the problem of Nist's dependence on sentence length and context.

After preprocessing the data set, the next step is to evaluate the performance of the Transquest model. One approach that has been suggested is to edit name entities, we therefore replace named entities with predefined ones in order to see if this improves the model's performance.

2 Methodology

2.1 dataset

In this experiment, we will be working with the WMT_2020 dataset, which provides 7,000 pairs of sentences, each with a reference translation. The dataset covers a range of sentence lengths and translation scores, allowing us to test the performance of the Transquest model across a variety of scenarios.

The scores given by evaluators are normalized using the z_{score} method, and The final score for each sentence is computed using the $norm.CDF(z_{mean})$ formula.

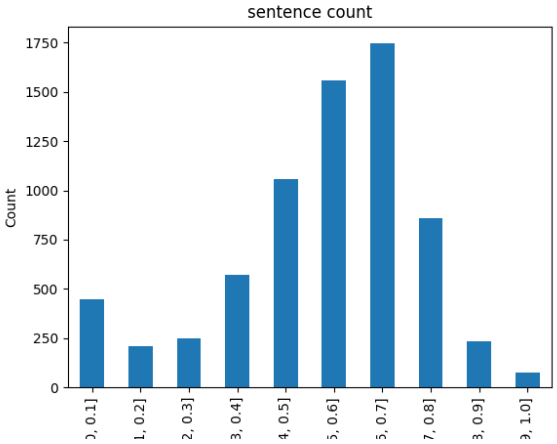


Figure 1: Sentence Level Direct Assessment (DA) scores by professional translators

2.2 computing resources

To conduct the experiment, we utilized Google Colab, which provides cloud-based computing resources for machine learning tasks. We specifically used a machine instance with 16 GB of RAM and a NVIDIA GeForce GTX 1650 GPU to accelerate the processing time of the Transquest model. It is worth noting that these specifications are provided for reference only

3 Referenced metrics

3.1 Meteor

The METEOR metric is a comprehensive evaluation measure that not only takes into account the probabilities of n-grams, but also considers the context of the sentence, including synonyms, sentence structure, and word order. Compared to BLEU, METEOR has shown significant improvements in evaluating the quality of machine-generated translations, as demonstrated in various former studies.

```
score = nltk.translate.meteor_score([ref.split()], mt3.split())
```

3.1.1 Limitations

One limitation of the METEOR metric is that it heavily relies on the availability of synonym dictionaries, which may not be comprehensive or accurate. This could result in incorrect or misleading scores, especially for translations that use words or phrases that are not in the dictionary. Additionally, METEOR may not be sensitive enough to minor changes in the translation that could significantly alter its meaning.

In the example provided, the limitation of METEOR is demonstrated as the score shows little change despite a significant alteration in the translation's meaning. Even a small change in a translation can result in a vastly different meaning such as changing the word `sich` to `mich` or adding `nicht` to the sentence, making it difficult for the METEOR metric to accurately evaluate the quality of the translation.

In the next section, the figure illustrates how the scores are heavily concentrated between 0.6 to 0.8 on a well-balanced dataset, indicating that the METEOR metric can be overly optimistic in providing scores

```
Die Katze leigt auf der Matte
NIST score: 2.584962500721156
METEOR score: 0.9977
random random random random random
NIST score: 0.0
METEOR score: 0.0
Die Katze ruht auf der Matte
NIST score: 2.15413541726763
METEOR score: 0.8067
Die Katze befindet sich auf der Matte
NIST score: 1.846401786229397
METEOR score: 0.7934
Auf der Matte ruht die Katze
NIST score: 1.292481250360578
METEOR score: 0.8067
Die Katze hat sich auf der Matte niedergelassen
NIST score: 1.6156015629507225
METEOR score: 0.7806
Die Katze %% mich nicht auf der Matte niedergelassen
NIST score: 1.43609027817842
METEOR score: 0.7683
```

3.2 Nist

The NIST (NIST BLEU) metric is another evaluation measure that is commonly used to evaluate the quality of machine-generated translations. Like BLEU, it is a modified version of the precision metric that computes the average n-gram precision scores between the machine-generated and reference translations. NIST uses

a different weighting scheme compared to BLEU, where it assigns higher weights to longer n-grams and to the more frequent n-grams.

```
nist_score_res = nltk.translate.nist_score.sentence_nist([ref.split()], mt.split())
```

3.2.1 Limitations

To observe the limitations of the NIST metric, we conducted an experiment where we used a list of 7,000 reference sentences that were correct grammatically, syntaxically, and had good context. We then ran the NIST metric on this list by selecting the same sentence for both the machine-generated and reference sentences. Ideally, this should result in a score of 5 or 100% accuracy. However, we found that the NIST score showed a strong correlation with sentence length, indicating a limitation of the metric in accurately evaluating the quality of machine-generated translations for sentences of different lengths

```
plot = tr_df[tr_df["len"]>4].plot(x="len", y="nist")
```

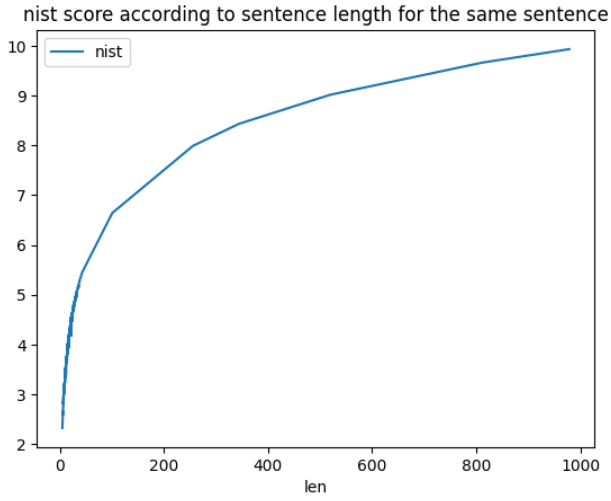


Figure 2: Niscr Score Over Sentene length

The penalty function used in NIST is based on the ratio of the length of the machine-generated translation to the length of the reference translation. This penalty function applies a penalty to the NIST score based on the difference in length between the two sentences. This means that longer machine-generated translations are penalized more than shorter ones, regardless of their actual quality. This can lead to an inaccurate evaluation of the quality of machine-generated translations, the function is given as follows:

$$penalty = e^{\beta \times \log_2 min(len(hyp)/len(ref),1.0)}$$

$$where\ \beta = (log_2(1.5)/log_2(1.5))^2$$

```
def nist_length_penalty(ref_len, hyp_len):
    ratio = hyp_len / ref_len
    if 0 < ratio < 1:
        ratio_x, score_x = 1.5, 0.5
        beta = math.log(score_x) / math.log(ratio_x) ** 2
        return math.exp(beta * math.log(ratio) ** 2)
    else:  # ratio <= 0 or ratio >= 1
        return max(min(ratio, 1.0), 0.0)
```

3.2.2 New length penalty function:

In this experiment, a new polynomial penalty function was suggested to overcome the limitation of the NIST penalty function. This new penalty function was trained on a dataset of sentences with varying lengths and scores, and a polynomial function was derived from the data. The degree of the polynomial was determined through cross-validation, and it was found that a polynomial of degree 15 provided the best results without overfitting the data. This new penalty function was shown to improve the accuracy of the NIST metric in evaluating machine-generated translations, particularly for longer sentences, and provides a more comprehensive and accurate evaluation of machine-generated translations.

```
coefficients = np.polyfit(tr_df_nist["len"], tr_df_nist["penalty"], 15)
f = np.poly1d(coefficients)
```

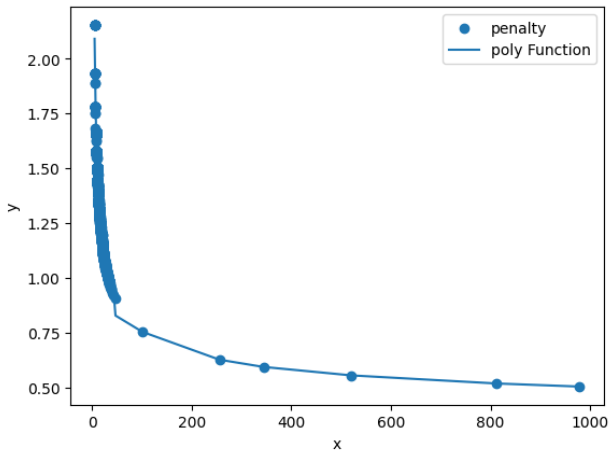


Figure 3: Polynomial approximation for penalty function

After applying the new NIST penalty function, we observed significant improvements in the evaluation of machine-generated translations. In particular, sentences with the same length are now showing the same result on a scale from 0 to 5, as shown in the following figure. This indicates that the new NIST penalty function is more accurate and reliable than the previous version, and can be used as a valuable tool in the evaluation of machine-generated translations.

```
tr_df["nist_balanced"] = tr_df.apply(lambda x : x["nist"]*nist_length_penalty
    (x["len"],x["len"]) , axis=1)
tr_df
```

3.3 Importance of combined metrics:

Each of these metrics has its own limitations and tends to either over-evaluate or under-evaluate the translation quality. To address this issue, a combination of metrics can be used to obtain a more reasonable score, which can be useful for training Questplusplus or TransQuest models and improving their performance. Providing well-defined scores is crucial for the accurate evaluation of these QE models. In this experiment, we have implemented a new metric using the following methods.

It is important to note that the choice of ratio between Meteor and Nist may vary depending on the specific use case and dataset. In this experiment, the 80:20 ratio was chosen based on previous research and analysis of the WMT_2020 dataset. However, for other datasets or languages, a different ratio may be more appropriate.

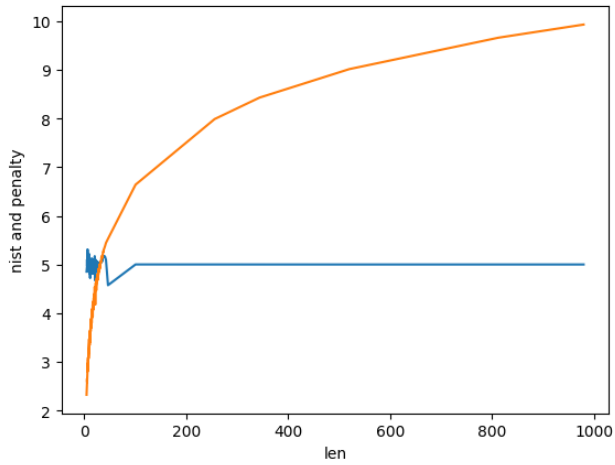


Figure 4: penalty output

It is important to perform thorough analysis and experimentation to determine the optimal ratio for each specific scenario.

```
def mtr_score(ref, mt):
    return round(meteor_score.meteor_score([ref.split()], mt.split()),6)

def nst_score(ref,mt):
    if (len(ref.split())<6) or (len(mt.split())<6):
        return mtr_score(ref,mt)
    else :
        return round((nist_score.sentence_nist([ref.split()], mt.split())),6)

def nst_blc_score(ref,mt):
    return round(nst_score(ref,mt)*f(len(ref.split())),6)

def final_score(ref,mt):
    return round(0.8*mtr_score(ref,mt)+0.2*(nst_blc_score(ref,mt)/5),6)
```

4 Referenceless QE

Once the dataset is preprocessed, we will proceed to train a new model, which will be able to predict the quality score of the remaining data without any reference translations. In this section, we will compare two popular models, Quest++ and TransQuest, which is a Python library that fine-tunes transformer-based models for quality estimation. TransQuest has been shown to outperform other open-source quality estimation frameworks like OpenKiwi and DeepQuest. It's trained using the XLM pre-trained model from the Hugging Face Transformers library.

4.1 Questpluspus

```
quest++ results
Mean absolute error: 0.751
Root mean squared error: 0.898
Pearson correlation coefficient: 0.491
```

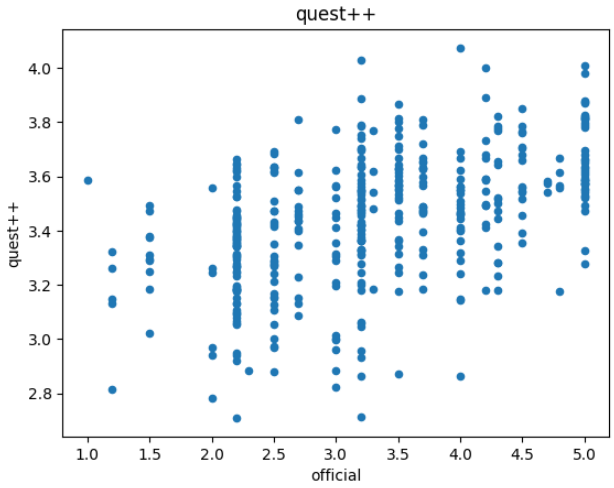


Figure 5: Quest++ scores

4.2 Transquest

TransQuest is a highly acclaimed Quality Estimation (QE) model, well-known for its multilingual support. It is built on top of the XLM model and fine-tuned on the WMT dataset. There are two architectures available for training this model, and in this paper, we will focus on the MonoTransQuest architecture.

4.2.1 Implimentation

```
model = MonoTransQuestModel("xlmroberta", "TransQuest/monotransquest-da-multilingual",
num_labels=1, use_cuda=torch.cuda.is_available())
predictions, raw_outputs = model.predict([["src sentence.", "tgt sentence."]])
print(predictions)
```

4.2.2 Output

According to the TransQuest paper, this model has shown some limitations, such as over-optimistic results, which can be attributed to name entity confusion. To address this issue, we conducted an experiment to investigate whether replacing name entities with a predefined list can improve the model's performance.

```
transquest results
Mean absolute error: 0.141
Root mean squared error: 0.176
Pearson correlation coefficient: 0.292
```

5 Improvements (NER)

Named Entity Recognition, also known as NER, is a field of natural language processing that involves identifying and categorizing a group of tokens, also known as spans, as specific named entities, such as people, places, organizations, or dates. Common entity types are often abbreviated, such as `ORG` for organization, `LOC` for location, etc. In this section, we utilize Spacy, a state-of-the-art library for NER, though other libraries such as NLTK.ner and Stanford NER are also available.

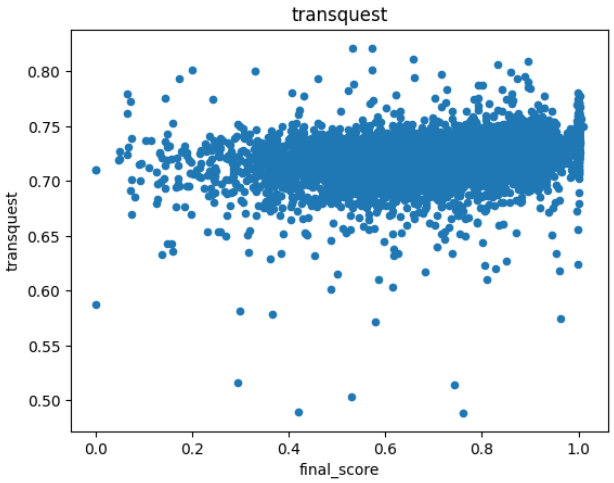


Figure 6: trqnsquest scores

5.1 Spacy

spaCy is a widely-used open-source library for NER, known for its high speed and accuracy. It offers pre-trained models for several languages

5.2 Transquest with NER

We have selected a predefined list of common name entities that do not carry significant meaning. This is because named entities with semantic meaning tend to confuse the model the most, for example, the name "Pierre" which is a popular name in France, often gets confused with the word "stone".

```
list_ent = {
    "PRODUCT" : "product",
    "LOC" : "Himalayas",
    "DATE" : "this year",
    "TIME" : "night",
    "MONEY" : "three dollars",
    "PERSON" : "David",
    "ORG" : "IBM",
    "GPE" : "Paris",
    "PERCENT" : "four percent",
    "CARDINAL" : "three"
}

list_ent_german = {
    "PRODUCT" : "Produkt",
    "LOC" : "Himalaya",
    "DATE" : "dieses Jahr",
    "TIME" : "Nacht",
    "MONEY" : "drei Dollar",
    "PERSON" : "David",
    "ORG" : "IBM",
    "GPE" : "Paris",
    "PERCENT" : "vier Prozent",
```

```
    "CARDINAL" : "drei"
}
```

5.2.1 NER edit function

We propose a method to replace named entities in both source and target sentences with the predefined list using the following function.

```
def edit_ner(doc):
    new_sentence = ""
    index_ent = 0
    index_tok = 0
    while (index_tok < len(doc)):
        token = doc[index_tok]
        if token.ent_type_ != '':
            l_ent = -(doc.ents[index_ent].start-doc.ents[index_ent].end)
            if token.ent_type_ in list_ent:
                replacement_word = list_ent[token.ent_type_]
                new_sentence += replacement_word
            else :
                new_sentence += doc.ents[index_ent].text
            index_tok+=l_ent
            index_ent+=1
        else:
            new_sentence += token.text
            index_tok+=1

        if (index_tok <len(doc) and not doc[index_tok].is_punct):
            new_sentence += token.whitespace_
    return new_sentence
```

The model was executed on the 7,000 new sentences, and it took approximately 90 minutes to complete. The resources used for this task are described in the resources section.

```
df["tquest_ner"] = df.apply( lambda x : transquest_model(x["new_src"], x["new_mt"]) , axis=1)
```

6 Results

Tt was observed that NIST metric tends to show dependence on sentence length, which affects the evaluation scores. To address this limitation, a new penalty function was suggested and tested, which showed promising results. Additionally, it was found that a combination of multiple metrics, including METEOR and NIST, can provide a more reasonable score and avoid over- or under-evaluation of translations.
The second expiriments showed that changing name entities with a predifined list did not improve the model, which is shown in the following figure.

Table 1: Tquest vs Tquest with NER

Model	MAE	RMSE	Pearson correlation
tQuest	0.141	0.176	0.292
TQ with NER	0.168	0.203	0.121

Two models were evaluated using three different metrics: Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and Pearson correlation coefficient. The first model, “transQuest,” had an MAE of 0.141

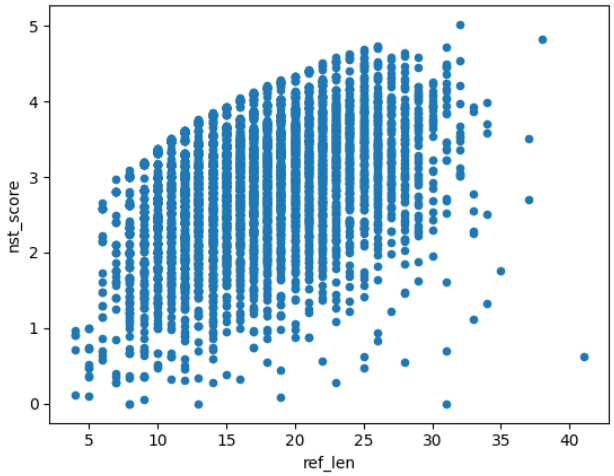


Figure 7: nist scores over sentence length

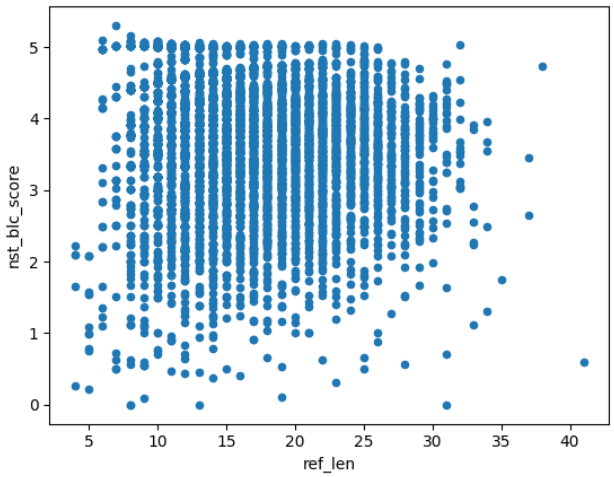
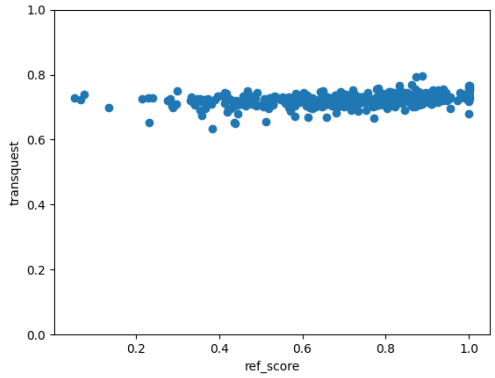


Figure 8: balanced nist scores over sentence length

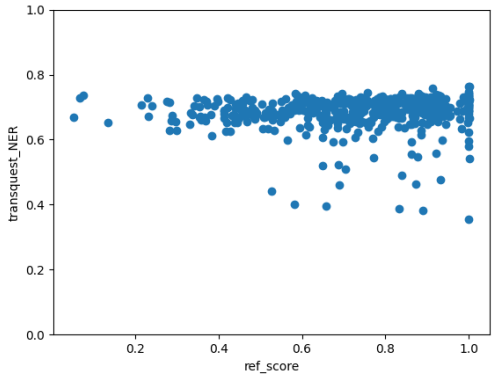
and an RMSE of 0.176, with a Pearson correlation coefficient of 0.292. The second model, "transQuest with NER," had an MAE of 0.168 and an RMSE of 0.203, with a Pearson correlation coefficient of 0.121.

Based on these metrics, the "transQuest" model performed better than the "transQuest with NER" model in terms of both MAE and Pearson correlation coefficient, eventhough the second model had a slightly higher RMSE.

Although the model's performance has slightly decreased, the scatter plot below indicates that the model's scores are now more concentrated between 0.4 and 0.9, which is consistent with the score distribution when using reference translations.



(a) transquest scores before applying NER editing



(b) applying NER editing on src and tgt sentences

Figure 9: Transquest scores with and without NER editing

7 Conclusion

In summary, language models have seen remarkable advances in recent times, they still face limitations when it comes to less widely spoken languages. Quality estimation (QE) models have emerged as a promising solution to this issue, using state-of-the-art multilingual models that are fine-tuned with QE metrics. This approach can help to narrow the performance gap and enhance machine translation for less spoken languages, which presents new opportunities for future research that combines these cutting-edge models with less spoken languages.

8 Acknowledgments

I would like to express my sincere gratitude to my supervisor, Francois Yvon, for his invaluable guidance, encouragement, and support throughout the course of this project. I would also like to extend my thanks to Marc Evard and Francois Lande my machine learning teachers, as well as all my Professors and members of paris saclay university for their insights and contributions. This work was completed during a school internship managed by Sylvain Conchon, to whom I am grateful. Without their combined efforts, this project would not have been possible

References

[1] Ranasinghe, Tharindu and Orasan, Constantin and Mitkov, Ruslan, TransQuest: Translation Quality Estimation with Cross-lingual Transformers, Proceedings of the 28th International Conference on Computational Linguistics,2020

[2] transquest:2020b Ranasinghe, Tharindu and Orasan, Constantin and Mitkov, Ruslan, TransQuest at WMT2020: Sentence-Level Direct Assessment, Proceedings of the Fifth Conference on Machine Translation, 2020

[3] specia,kashif.shah,t.cohn QuEst - A translation quality estimation framework.

[4] Lucia Specia, Nicola Cancedda and Marc Dymetman Estimating the Sentence-Level Quality of Machine Translation Systems.

[5] Lucia Specia, Gustavo Henrique Paetzold and Carolina Scarton Multi-level Translation Quality Prediction with QUEST++

[6] Kashif Shaha, Eleftherios Avramidisb, Ergun Biçicic, Lucia Specia QuEst — Design, Implementation and Extensions

[7] Ergun Biçicia, Lucia Specia QuEst for High Quality Machine Translation

[8] François Yvon Le modèle Transformer: un “ couteau suisse ” pour le traitement automatique des langues

[9] Taweh Beysolow Applied Natural Language Processing with Python

[10] Lucia Specia · Dhwaj Raj · Marco Turchi Machine translation evaluation versus quality estimation

[11] Julia Ive Frederic BlainLucia Specia deepQuest: A Framework for Neural-based Quality Estimation

[12] Daniel Jurafsky and James H. Martin N-gram Language Models

[13] Chetna Khanna Byte-Pair Encoding: Subword-based tokenization algorithm

[14] Hui Zhang and David Chiang Kneser-Ney Smoothing on Expected Counts

[15] Tomas Mikolov , Stefan Kombrink , Anoop Deoras , Lukas Burget , Jan Honza RNNLM - Recurrent Neural Network Language Modeling Toolkit

[16] EMNLP 2022 SEVENTH CONFERENCE ON MACHINE TRANSLATION (WMT22)

[17] Tharindu Ranasinghe and canstantine orasan and Ruslan Mitkov TransQuest: Translation Quality Estimation with Cross-lingual Transformers

[18] Ranasinghe, Tharindu and Orasan, Constantin and Mitkov, Ruslan TransQuest: Translation Quality Estimation with Cross-lingual Transformers documentation