

# Using Django and MongoDB to Develop a Translation Quality Checking Tool

Subhashree Chowdhury, Ryan Egbert, Poorna K Narasimhan, Suyash Sukthankar, Matthew A. Lanham

Purdue University, Department of Management, 403 W. State Street, West Lafayette, IN 47907  
chowdh30@purdue.edu; regbert@purdue.edu; krish212@purdue.edu; ssuktha@purdue.edu;  
lanhamm@purdue.edu

## ABSTRACT

*As literacy continues to spread throughout the world, more and more documents are being translated into languages that did not have prior access to such information. To ensure accurate and complete translations, this project provides a system through which the quality of a translation can be determined. In order to ensure ease of access, a visual interface was built to allow users the ability to check the quality of translations. Further, users are able to view previous versions of a translation and are given insights on how to improve certain aspects of a translation. Currently, there are no full-fledged web applications that provide insights into translation similarity, comprehensibility, and readability quality checks through an interactive interface. To meet these specifications, the application was built using the Django web framework, due to its ability to enable rapid development. The steps in the development process included high-level and low-level design, database implementation, front-end development, and model implementation.*

**Keywords:** Machine Translation, Quality Check, Semantic, Similarity, Readability, Comprehensibility, Django, Web framework

## 1 INTRODUCTION

Currently, all throughout the world, people across different parts of the world speak several distinct languages - 7,151 according to Ethnologue (2022, March 11). Very little communication happens without language. Yet, the barrier is not language, but the inability to transfer information between languages. Owing to this, translation of text in local and regional languages becomes vitally important for the correct transmission of information and insights. For example, books and other literature are translated into various languages for diverse audiences by multiple translators. However, due to the inherent differences in languages and manual errors by the translators, the quality of translation becomes less and less reliable (Rahmatillah, Kartini, 2016). In many cases, the quality is of vital importance (for example in the translation of a medical procedure) where any misunderstanding could lead to lawsuits or even fatal mistakes. In other cases, the meaning of the original text is symbolic or poetic and it is considered important for this to be retained in the translation. One way to combat the chance for poor quality translations is to build an application for users to assess the quality of their own translations using machine learning and artificial intelligence techniques. The application aims to cover three types of functionalities: semantic similarity, comprehensibility, and readability. The semantic similarity metric displays a numerical representation of the similarity between two texts. Comprehensibility highlights possible questions to the parts of the translated text and flags if the inferred answer is correct. Readability conveys to the user regarding the level of text complexity for various components of its translated text. Based

on the user's needs, the selection between the functionality and score measure needed for the assessment is made available.

There are three types of users that will have access to the application: translators, consultants, and administrators. Translators are given a source text and tasked with translating the text into a target language. Consultants are given both a source text and its corresponding translation and are responsible for determining the areas and topics that need attention. Administrators have access to all translations and metrics that both translators and consultants create, with the additional ability to grant access to other users.

The models used in the quality checking had been pre trained for a specific purpose that was leveraged to fit the needs of the application.

The application is built using Django (Version 3.2.12) [Computer Software] (2021) which is a framework in python. Django is user-friendly and has a vast suite of in-built features. Usage of Django enables the implementation of all features and functionalities without relying on anything else. The application also utilizes MongoDB (Version 5.1) [Computer Software] (2021).

During the development process, the Agile methodology of software development was implemented. Development was accomplished in four months divided into two week sprints. The implementation followed requirements provided from the stakeholders. During the development process, testing and validation was necessary before adding any code into production. GitHub (2021) is used as the version control mechanism which makes the errors easy to identify and rectify on a timely basis.

## **2 LITERATURE REVIEW**

A considerable amount of research has been done on the benefits and drawbacks of different web framework technologies. Various research papers have been studied extensively to conclude a web framework to be used for building this application. The options considered were Streamlit, Base Ten Systems, Flask, and Django. Each framework was assessed and explored thoroughly, along with its various versions to decide the ultimate technology to be used which was also most compatible with the other software and hence suitable for building this application.

The final decision was to leverage Django 3.2.12 for the development due to its multiple advantages. Firstly Django 3.2.12 was compatible with MongoDB. Additionally, in Medium's blog posted on "Why choose Django over Flask", Django has more time to gather numerous extensions, plugins and third-party apps covering a wide range of needs than Flask. Unlike Flask, Django comes with a built-in bootstrapping tool — django-admin — which enables developers to start building web applications without any external input. Flask is more suited to smaller, less complicated applications, while Django is designed for larger, more complex, and high-load applications. Secondly, as observed in many publications it has several advantages over Streamlit and other Python frameworks.

A study titled "A Generic Review of Web Technology: Django and Flask" delves deep into the different web frameworks, their history, future scope, relevance, and analyses the threats they face. The study goes on to evaluate how the different technologies deal with the various threats they face. The study analyses all the technologies in the full stack from Front end web frameworks to database technologies and derives a conclusion regarding the pros and cons of using each technology, especially Django and Flask.

In the study titled “RESTful Web Service for Madurese and Indonesian Language Translator Applications on Android Devices” there has been an attempt to create an application that would run on android devices and can be used as an Indonesian to Madurese Translator. The application uses RESTful web service with the data being stored and transmitted in a JSON format. The architecture consists of two applications, one containing the web service application running on the client-side and another containing the android-based translation application running on the client-side. The tests indicated that the response time for the application is related to the amount of data received and that the response time for the android application is lesser than that of the web application already available.

In another study titled “Investigative Study towards the Development of Mobile Language Translator” the aim was to conduct an analysis towards the development of a mobile language translator that is used to translate between English to Mandarin and Tamil and vice versa. The aim of this application is to facilitate a platform for tourists to ease their language barrier issues. The application enabled the translation of some common phrases that tourists used and provided pictorial representation wherever necessary to help people. In developing this application, the Object-Oriented Analysis and Design (OOAD) with Unified Software Development Process (USDP) is used.

According to “Translation errors in the process of translation,” translation errors mostly result from the non-equivalence between the source and target languages. There are many aspects involved in translating, not only translating the text meaningfully but also retaining the cultural aspects and the original essence of the source text.. It is not a simple task to do. If the expression or the text is hard to understand even to translators themselves, the function of translation then fails to fulfill. The ways to overcome one error will influence the solutions for other errors. Thus, figuring out the quality of the translation, understanding the issues in it and figuring out the readability and comprehensibility.

### 3 DATA

In order to fully test the reliability of the application, data that had been translated with a high level of accuracy was necessary. We obtained this data from SIL International (2022) and the Bloom Library (2022) specializing in linguistics research. The data in question was pulled from a collection of books with information on COVID-19 translated in dozens of different languages.

Originally in a JSON format, we proceeded to transform the data into CSV files. Each sentence from the text is given in both the source text as well as the translated text.

Table 1: Data dictionary for test translations

Variable	Type	Description
<b>lang1</b>	Text	The text in the first language (generally the source language)
<b>lang2</b>	Text	The text translated into the second language
...	...	
<b>langn</b>	Text	The text translated into the n <sup>th</sup> language

Converting the data into this format helped us easily traverse the dataset, leading to simple validation. Being able to go row by row, feeding each instance of the text into language models proved to be beneficial.

## **4 METHODOLOGY**

### **4.1 Requirements Gathering**

Prior to beginning development, requirements were made for what the final product must contain. Many of the requirements detailed specific functionality components while others focus on the look and feel of the application.

Many of the requirements provided were standard, detailing the register and login process, navigation between pages, and allowing users to access previous translations. Other requirements, however, gave detailed information on how much time should pass (or how many clicks a user would make) in order to arrive at specific information. Visual and accessibility requirements were made for the overall look of the application. Other requirements indicated that a user should have a large amount of flexibility in determining what information should be available to them.

Example functionality requirements:

- See detailed score information in less than two clicks (or by hovering)
- Only run the metrics I select
- See information on aggregated scores from across multiple translations of the same reference
- Do comparisons of two translations based on the same references before and after changes

Example aesthetic requirements:

- A color palette that allows major differences to be easily distinguished for color-blind users
- Increase the size of visual elements
- Display major information identifiable with a screen reader
- WCAG 2.1 Level A criteria met at minimum

### **4.2 High Level Design**

Following the process of obtaining and understanding requirements, the application was in a process to be designed both visually and abstractly. Page designs were prepared for each of the three metrics that would be used: semantic similarity, comprehensibility, and readability. The visual design of the pages was provided to the stakeholders with approval to continue the process.

### **4.3 Prepare Templates/Design UI pages**

Following approval from the stakeholders, static templates were generated from the high level design. These templates were made with very minimal HTML and CSS to reduce the complexity of the project. Following a quick development, the templates were once again presented to the stakeholders with the intention of adding complexity and functionality.

### **4.4 Integrate templates with Django**

The Django framework uses the Jinja2 templating engine (*Templates*) which allows developers to programmatically generate dynamic web pages. Jinja2's simple syntax, paired with Django's liberty on the backend, create an easy and compact way to turn static templates into dynamic web pages.

#### **4.5 Integrate metric models**

The final step in the development process was to integrate the metrics into the backend of the application. Each quality metric used a different algorithm to compute some numerical score of quality. (for more info, see Metrics) These scores were then displayed to the user in a visual way, giving the user the ability to see which areas and categories of a translation required the most attention.

#### **4.6 Test application**

The final product was tested with data that had been provided by the stakeholders. This data was composed of the text from many different pamphlets and brochures that had been translated into a variety of languages.

To test that the application would detect a poor translation, the development team generated data that contained many translations that ranged from high quality to low quality.

### **5 MODELS AND METRICS**

There are three metrics used to determine the quality of a translation: semantic similarity, comprehensibility, and readability.

#### **5.1 Semantic Similarity**

Semantic similarity measures the “closeness” of two different pieces of text. This process may be accomplished in the same language or even across multiple different languages.

To determine semantic similarity, it is first required to convert sections of text into a numerical format. This process is called sentence embedding. While there exist many different models that can be developed and used in order to embed sentences into a numerical representation, the model that was selected for this project was the “Language-agnostic BERT Sentence Embedding” pretrained model (<https://arxiv.org/pdf/2007.01852.pdf>). This model is a multilingual sentence embedding model that can convert text in over 100 different languages.

In the application, a user submits two different texts, the source text and the translated text. The application will then perform sentence embedding on both texts, resulting in two representative numerical vectors. These vectors are then compared with one another, and a score is generated by regarding the similarities or differences between them. Scores range from 0 to 5, 0 being completely dissimilar and 5 being semantically identical.

#### **5.2 Comprehensibility**

Complete comprehensibility is established when a piece of translated text contains the same important information as the source text. The way comprehensibility is determined is by using a series of question-answer pairs. By “asking” several questions, a model can determine if important and specific information is being captured by a piece of the translated text.

The model that is used to determine the comprehensibility score of the text was developed by Google called “BERT (base-multilingual-cased) fine-tuned for multilingual Q&A”. (Google Research, 2019). This model contains question and answer information on over 100 different languages.

Google’s model takes in as input the text to search and the question that is to be answered. In response, the model provides several key points of information: the answer to the question derived from the text, a probability score, and the exact location of the answer in the text.

While a user is submitting a translation for scoring, they may submit questions and provide the expected answers to those questions as they pertain to the translated text. While the application is scoring the text, the questions will be machine translated into the target language and fed into the model. The model will return the answer that was derived from the text. Using this answer and the answer provided by the user, a similarity score can be determined. If the similarity score surpasses a certain threshold, the question will be considered answered and correct. The overall comprehensibility score will be based on the number of questions that were provided in relation to the number of answers that were correctly identified.

This process may have its own hurdles to overcome, depending on the language of the translated text. If a question is unable to be translated, the model will be unable to provide an accurate comprehensibility score.

### **5.3 Readability**

There are many different algorithms by which a readability score can be calculated. One of the most common of these algorithms is the Flesch-Kincaid readability score (Kincaid, J. P., et al., 1975). The Flesch-Kincaid readability score uses the number of words per sentence and the number of syllables per word to determine the complexity (and thus readability) of a piece of text.

While using the number of syllables in each word would be ideal, that information is not widely available for many different languages. Instead, the algorithm being used in the application determines the number of words per sentence and the number of *characters* per word to indicate the readability of a sentence. The idea is that text that uses longer words in its sentences tends to be more complicated, and therefore more difficult to read.

As opposed to the two previous metrics, there is no way to determine a “good” or “bad” readability. For example, a scientific paper will have a higher readability score than a children’s book, but that does not mean the paper is any worse than the children’s book. All the readability score indicates is the level of education or domain knowledge required in order to fully understand a piece of text.

Rather than show whether or not a translation has “good” readability, the application indicates where there are pieces of text where the readability varies from the overall readability of the text. This will indicate to the user the areas that may have too high or too low a readability relative to the rest of the text and encourage the user to adjust based on the feedback.

## **6 FUNCTIONALITY**

During development of the application, many different features were added to increase ease of access to the wide range of different users. After base requirements had been met, some aspects of the application needed to be improved or updated to give the user maximum quality. From there, maintenance and improvements were added to reach peak functionality.

There are four main features of the application.

## 6.1 Register and Login

Allowing users to register and login to the application provides a system through which users can keep track of their own profiles. User settings, preferences, and past translations are saved in a database, giving users the ability to use the application with freedom.

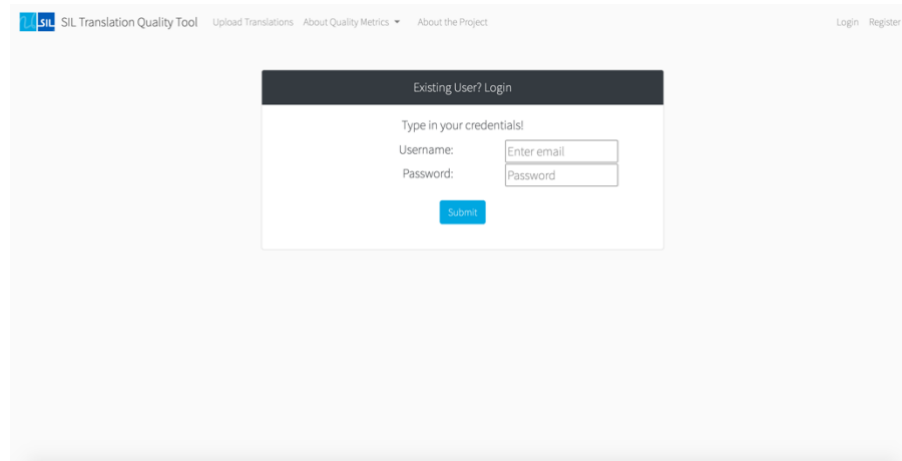


Figure 1. Login page

## 6.2 Upload Translations

Once registered, users must be able to upload their translations, whether via a file upload or manual text entry. Files that are uploaded are required to follow a specific structure. Files must be a CSV with four populated columns: source text, translated text, question, and answer. The question and answer fields are specific to the comprehensibility metric.

source	translated	question	answer
Ondanks wat je leraar je misschien heeft verteld, is er een verkeerde manier om een ÄsÄslasso te hanteren.	Despite what your teacher may have told you, there is a wrong way to wield a lasso.	Is there a wrong way to wield a lasso?	Yes
Ze hadden dringend een andere drummer nodig aangezien de huidige alleen bongo's kon spelen.	They desperately needed another drummer since the current one only knew how to play bongos.		
Je weet niet zeker of je hem wel of niet moet vertrouwen, maar erg dankbaar dat je een coltrui droeg.	You're unsure whether or not to trust him, but very thankful that you wore a turtle neck.		
Als je van tonijn en tomatensaus houdt, probeer de twee dan te combineren, het is echt niet zo erg als het klinkt.	If you like tuna and tomato sauce, try combining the two, it ÄsÄs really not as bad as it sounds.		
Hij dronk het leven voordat hij het uitspuugde.	He drank life before spitting it out.		
Op dat moment leerde hij dat er bepaalde delen van het lichaam zijn die je	It was at that moment that he learned there are certain parts of the body that you		

Figure 2. Sample CSV layout

Users are also able to select which metrics to include in their translation processing. Once the user clicks “Submit”, the text will begin processing. Processing time will increase based on the number of metrics selected and the length of the source text.







Figure 4. Sample results page

## 6.4 Past Translations

Users can also review their past translations. This would benefit if a translator had been working on a specific translation over time and wanted to look over which areas needed the most work. Translations can be updated with more questions that add to the comprehensibility score of the translation.

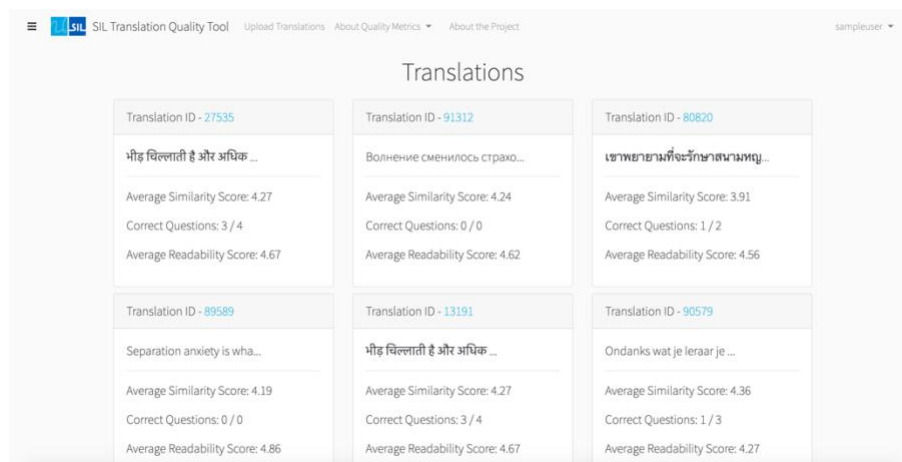


Figure 5. Past translations page

## 7 CONCLUSIONS

The Django framework provided a solid foundation for rapid development of a translation quality checking tool. The tools that were provided, including the flexibility of the Python programming language, provided an environment through which a front-end application could be easily developed. Additionally, the structure of MongoDB databases provided a simple, yet effective structure that could be utilized to store, manage, and process large amounts of unstructured data. Many pretrained models were utilized in order to check the semantic similarity and comprehensibility between two separate texts, as well as the readability of a translated document. This application can provide all types of users – with roles such as translators, consultants, and administrators – ease of access and many visual metrics of the efficacy and quality of their translations.

## REFERENCES

1. Rahmatillah, K. (2013). Translation errors in the process of translation. JEE, Journal of English and Education, 7(1).
2. Django (Version 3.2.12) [Computer Software] (2021). Retrieved from <https://docs.djangoproject.com/en/4.0/releases/3.2.12/>
3. MongoDB (Version 5.1) [Computer Software] (2021) Retrieved from <https://docs.mongodb.com/manual/release-notes/>
4. Idris, N., Foozy, C. F. M., & Shamala, P. (2020). A generic review of web technology: Django and flask. International Journal of Advanced Science Computing and Engineering, 2(1), 34-40.
5. Pramudita, Y. D., Putro, S. S., Wahyudi, R. N., Suzanti, I. O., & Solihin, F. (2020, October). RESTful Web Service for Madurese and Indonesian Language Translator Applications on Android Devices. In 2020 6th Information Technology International Seminar (ITIS) (pp. 203-206). IEEE.
6. Fong, S. L., Elfaki, A. O., Johar, M. G. M., & Aik, K. L. T. (2012). Investigative study towards the development of mobile language translators. International Journal of Digital Content Technology and its Applications, 6(22), 11.
7. Annenkov, & Cherkashin, E. A. (2013). Generation technique for Django MVC web framework using the stratego transformation language. 2013 36th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), 1084–1087.
8. Yu, & Yang, W. (2019). The Analysis and Design of System of Experimental Consumables Based on Django and QR code. 2019 2nd International Conference on Safety Produce Informatization (IICSPI), 137–141. <https://doi.org/10.1109/IICSPI48186.2019.9095914>
9. Yusuf, Ahmed. (2018, September 25). Why choose Django over Flask. Medium Blog. <https://medium.com/@ahmedamedy/why-choose-django-over-flask-a287be9b22a4>
10. Ethnologue (2022, March 11). Languages of the World. Ethnologue. <https://www.ethnologue.com/>
11. SIL International (2022). Resources. <https://www.sil.org/resources>
12. The Bloom Library (2022). <https://bloomlibrary.org/>
13. Bentivogli, Luisa, et al. “Revising the Wordnet Domains Hierarchy.” *Proceedings of the Workshop on Multilingual Linguistic Ressources - MLR '04*, 2004, <https://doi.org/10.3115/1706238.1706254>.
14. Feng, Fangxiaoyu, et al. “Language-Agnostic BERT Sentence Embedding.” *Google AI*.
15. Kincaid, J. P., et al. “Derivation of New Readability Formulas (Automated Readability Index, Fog Count and Flesch Reading Ease Formula) for Navy Enlisted Personnel.” 1975, <https://doi.org/10.21236/ada006655>.
16. Google Research, BERT, 2019, GitHub repository. <https://github.com/google-research/bert/blob/master/multilingual.md>.
17. *Templates*. Django Documentation. (n.d.). Retrieved March 11, 2022, from <https://docs.djangoproject.com/en/4.0/topics/templates/>