Claire Lueking
Professor Guinn
CSPB 4380
19 April 2025

Machine Translation: English to German

Abstract:

This project created four models, two basic and two custom, to help facilitate the translation of sentences from English to German. A custom, extremely simplified tokenizer was made to tokenize sentences and words to aid in the translations as the planned tokenizer (spaCy) was a little too complex and heavyweight for the goals of the project tokenizer. Several rows of test sentences were tested after the models were trained and included a few custom short sentences for manual (human-based) translation checking based on a combined knowledge of intermediate German through myself and a friend. The number of rows run out of the several million in the dataset was only a small subsection (150,000 rows). This was such a small sample from the set due to local machine constraints and storage issues using an online provider to allow for a larger sample set and faster runtime. In the end, the custom models did better than the basic models, with the custom unigram model outperforming the custom bigram model in translations.

Introduction:

The problem that was addressed was trying to create n-gram based models to be comparable or close to comparable to Google Translate's translation model based on the BLEU scores of all the models involved. This is an interesting and useful problem because German is an extremely complex language with its sentence structure, grammar, and vocabulary. An example of this would be that there are six different ways to say "the" in German based on masculine, feminine, plural, and more rules. This makes it challenging for models to provide good-quality translations with low error rates. One of the best models out there is Google Translate, which has an approximate BLEU score of 4.71 out of 10 (causeweb.org 2023). This score indicates that the translations outputted by the model are about 47.1% correct. Overall, the goal is to make n-gram (unigram and bigram) models with BLEU scores as close as possible to Google Translate's BLEU score.

Related Work:

Similar work has been done on this subject of machine translation. One example of this would be a paper on n-gram based and phrase-based statistical machine translation by Durrani et al. in 2015. The model developed was titled the Operation Sequence Model (OSM) that integrates n-grams and phrase-based translations into the model in order to get better performance in translations that just using one version of translation over another. Key features of this model are that there are small translation units that consider source and target language word groupings, avoids assumptions that phrases are all independent of one another, uses dynamic graph searching to help allow for some flexibility in decoding large translations, and more. In the end, this model outperformed phrase-based and n-gram systems that were state of the art across standard translation tasks by limiting individual model limitations by combining a few types of models into one (Durrani et al. 2015).

Another example of related work is Fredrico, Marcello, and Mauro Cettolo's 2007 paper regarding the use of n-grams in statistical machine translation models, but as the model grows, memory requirements become too much to reasonably handle with standard computers. The solution proposed was to optimize n-grams by partitioning the words into subsets based on the frequency of the word to help find n-grams in smaller chunks for processing in memory. Additionally, to efficiently store and retrieve n-grams, a dynamic prefix tree data structure was created to help facilitate fast access to n-gram related probabilities, back-off wights, and so on. The last part of the solution was to use memory mapping for very large language models (LLMs) to help sections of the model to be loaded on-demand from memory. Overall, this model helped to reduce memory usage by 50% compared to more traditional models, but decreased translation speed by 44%, which was acceptable by the authors due to the significant savings in memory (Fredrico, Marcello, and Mauro Cettolo 2007).

These examples relate to the current model being proposed by their use of n-grams and the aim to optimize sections of the n-gram-based models to improve the efficiency and accuracy of the translations. The model that was proposed for the project ends up optimizing the model by messing with weights related to word frequency, similar to what was mentioned in the second example paper. Overall, these papers inspired me to do some digging into ways to optimize the n-gram models for better translations overall.

Data:

The dataset used was taken from Mohammed Lofty's Kaggle that was titled "WMT 2014 English-to-German". The structure and size of the dataset includes 4.5 million parallel sentence pairs in English and German. It also came partially preprocessed into .csv files. Further cleaning and preprocessing took place to clean up spacing, punctuation, keeping common German characters that aren't in the English alphabet, and turning NULL values into empty strings for ease of tokenizing and training data. The data was limited to long and complex sentences for the most part, so when testing basic sentences, it may not get exact translations as easily as more basic datasets.

Methodology:

The basic plan included a baseline of cleaning, preprocessing, and tokenizing the data. Then the models were created and tested. For cleaning and preprocessing, I used Python libraries pandas, re, numpy and collections' defaultdict. These helped to load the dataset, keep and get rid of certain characters (i.e. German characters for keeping and get rid of certain spaces and punctuation), and aid in faster processing when building a tokenizer as well as processing stats quicker for results. I ended up building an extremely basic tokenizer as spaCy, an already existing tokenizer I was planning on using, was a little too complex and heavyweight for my specific needs in the project. The collections library was used to utilize defaultdict which creates default keys for missing values in a dictionary. The NLTK framework was utilized to import a library to help create ngrams via nltk.util nrgrams.

For setting up training and metrics, the sacrebleu library (specifically corpus_bleu) to aid in calculating standardized BLEU scores for the models, and sklearn.model_selection's

train_test_split to help take the data to use 80% for training and 20% for testing. In creating the models, I started out with basic bigram and unigram models, but found they didn't work very well, hence I added custom unigram and bigram models that ended up being better than the basic n-gram models. I also briefly tried byte-pair encoding (BPE), but found out it was much harder to implement in a timely fashion that sticking with n-grams, so this pushed me towards developing the n-gram models more and scrapping the BPE version.

After the models were created and cemented as the path forward to work on, they were trained. The training included the file from Kaggle labeled "wmt14_translate_de-en_train.csv" that was cleaned according to the above data section, split into German and English sentences, and tokenized using the simple tokenizer as well as a few simple sentences that were easily translatable by humans for manual verification of the accuracy of the machine translations such as "Hello", "How are you?", "The government has lots of power to make decisions", and a few more. These basic sentences were mostly used as sample tests of the models when repeatedly running the data. At the end of testing the models, the evaluation metrics were calculated. First, the BLEU scores were calculated using corpus_bleu from the sacrebleu library, and then the accuracy and F1-score were calculated before printing this information out at the end of the code running cycle.

Results:

Model Scores Table

Model	BLEU Score	Accuracy	F1 Score
Unigram	0.4044	0.1121	0.1071
Bigram	1.4795	0.2176	0.246
Custom Unigram	3.1071	0.2801	0.2056
Custom Bigram	2.6473	0.2533	0.3116

Table 1. *The table illustrates each model and their scores (BLEU – the main metric as well as accuracy and F1 score for word for word translations.*

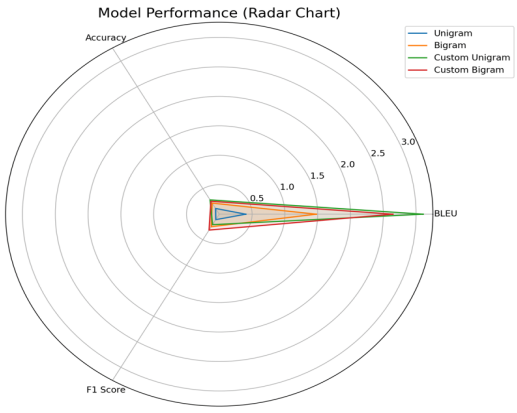
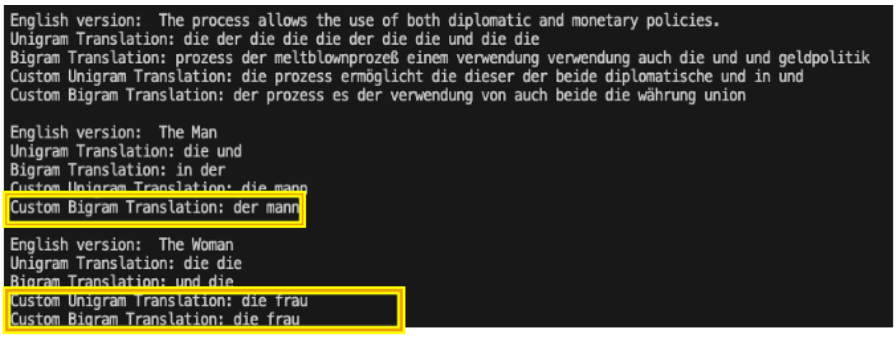


Chart 1. *This radar chart illustrates the models more visually than the table in terms of the best BLEU scores and their accuracy and F1 scores.*



Output 1. *This output shows the English version of the example sentence or phrase and the translations of all the models. The highlighted translations are a sample of direct translations from English to German.*

For the results, the model that performed the best with a BLEU score of approximately 3.17 out of 10 was the custom unigram model, followed by the custom bigram model at about 2.65 and the worst performing models were the basic bigram and unigram models, 1.48 and 0.40 respectively. If you look closely into the visualization, output, and the table, you will notice that the accuracy and the F1 score are low. This results from the number of improvements done to the models. The BLEU score was the focus of the project as the accuracy and the F1 score are not as reliable for machine translations overall. It is also important to note that the accuracy and F1 scores are for word-for-word translations whereas the BLEU score measures the overall translation. The BLEU score hold more weight due to the complexity of the metric that aligns with the complexity of languages, especially with the complicated language structure of German.

Discussion:

The findings were not what I hoped for as the accuracy and F1 score were too low for my liking and the BLEU score could have been closer to Google Translate's 4.71. What worked well was tuning the data for word frequency to penalize too common of words such as "the" and give more weight to other sort of common words to help get other words into the translation as opposed to having translations with only "the" and "and" in them. In the end, if I had to do this over, I would fix the issues I was having with Google Colab in order to be able to run more data faster as opposed to just being able to run on my local machine, which took a long time and I couldn't run as much data as I would have liked. I also would consider using a neural machine translation model (NMT) instead of n-grams for a slightly easier way to translate complex languages.

Conclusion and Future Work:

In summary, with the limited amount of time I had and the persistent difficulties, I think the BLEU scores for the custom models turned out well. In the future, I would expand the amount of data used for training and testing to improve translations in effort to capture more complex language issues such as grammar and sentence structure. Another direction this could be taken would be to specialize the model for a specific industry such as medical or legal, in order to focus on one area to improve translations for specific topic.

Works Cited

Causeweb.org. “A comparative Analysis of Translation Performance: ChatGPT vs. Google Translate.” *Causeweb.org*, 2023, https://www.causeweb.org/usproc/sites/default/files/usclap/2023-1/usclap%203010%20%20a%20comparative%20analysis%20of%20translation%20performance%20chatgpt%20vs%20google%20translate.pdf.

Durrani, Nadir, et al. "The operation sequence model—combining n-gram-based and phrase-based statistical machine translation." *Computational Linguistics* 41.2 (2015): 185-214.

Federico, Marcello, and Mauro Cettolo. "Efficient handling of n-gram language models for statistical machine translation." *Proceedings of the Second Workshop on Statistical Machine Translation*. 2007.

Lotfy, Mohamed. “WMT 2014 English-German.” *Kaggle.com*, 2024, http://www.kaggle.com/datasets/mohamedlotfy50/wmt-2014-english-german.