Project 14 - Variations in Construction Codes

Chi-kiu Lo
NRC-DT / Ottawa, ON
ChiKui.Lo@nrc-cnrc.gc.ca

David Minicola
NRC-CONST / Ottawa, ON
David.Minicola@nrc-cnrc.gc.ca

Shriya Vaagdevi Chikati
NRC-DT / Ottawa, ON
ShriyaVaagdevi.Chikati@nrc-cnrc.gc.ca

Abstract

Construction Codes of Canada are a set of safety requirements and guidelines to follow while constructing buildings and facilities. National Research Council Construction Research Center is responsible for facilitating the development and production of National Model Codes. NRC updates them once every five years and individual Provinces/Territories adopt these National Codes with differences. It is crucial to keep track of these variations to harmonize Codes. This paper introduces the approaches taken towards finding the variations between sentences in two documents: National Construction Codes and individual P/T Codes. It details the steps taken in preprocessing of the data and highlights the types of models used to find the variations, present the results, ending with a discussion of the future work.

1 Introduction

National Research Council Construction Research Center is responsible for facilitating the development and production of National Model Construction Codes that contain a set of objectives to follow while constructing a building or a facility. NRC CONST updates these Codes once every five years and the Codes are of four types: Building, Plumbing, Energy, and Fire.

Each Province/Territory adopts the National Codes with differences. Difference refers to any dissimilarity between the National and the P/T Codes and it can be of two types: Editorial and Technical (often referred to as a variation). An Editorial difference means that both sentences mean the same. A Technical difference means that the difference in the National and P/T Sentence has a material effect on the way that the building or facility is constructed, occupied, or designed.

An intergovernmental initiative that started in 2020 aimed at harmonizing these Codes, which includes the reduction of variations, to support interprovincial trade. This meant that the differences need to be tracked. The effort for tracking these variations was highly manual and it takes a considerable amount of time, effort, and resources to keep track of the variations since the National Codes alone have more than 14,000 sentences. It becomes even more complicated in cases where the P/T Codes do not follow the same numbering convention or order as the National Codes.

There are two versions of the Codes: 2015 and 2020. The variations for the 2015 data between the National Codes and the P/T Codes were tracked manually and the objective is to use AI and NLP techniques to identify them for the 2020 data.

We have different files for the 2015 data: variations files and excel files with all the sentences from the National and the P/T Codes in them. The 2020 data is in the form of XML files. We start by preprocessing the 2015 data.

The idea is to split the 2015 data into sets based on the Province/Territory and further split each of them into train and evaluation sets. Then, for each P/T, convert the full National and P/T sentences from the excel files into vectors/embeddings, compute the cosine similarity scores between the National sentence vectors and the P/T sentence vectors, and get maximum weight matchings (or the alignments). We then choose the optimal thresholds to keep or split the alignments based on the Alignment Error Rates calculated with respect to the train set and then use those thresholds on the alignments and get the final test Alignment Error Rate.

2 Dataset

There are two versions of the dataset: the 2015 version and the 2020 version. The variations for the 2015 version have been detected manually, which are in the form of various csv files. The sentences and the related information from both the 2015 National Codes and the 2015 P/T Codes are in excel files. The data for the 2020 data, for which the variations have to be detected, are in the form of XML files.

The columns of interest from the 2015 variations dataset are: National Sentence Text, P/T Sentence Text, Difference Type, Variation. The Difference Types and what they mean are as follows:

- National Only
  - The P/T has decided not to include the National sentence in their version of the Codes.
  - The P/T Sentence Text field is blank while the National Sentence Text field is not.
- P/T Only
  - The P/T has decided to add a new sentence not in the National Codes to their version of the Codes.
  - The National Sentence Text field is blank while the P/T Sentence Text field is not.
- Common Sentence
  - The sentences in the National Code book and the P/T Code book are a common sentence which could mean that, they have the same meaning or that they have a variation.
  - Both the National Sentence Text and the P/T Sentence Text fields are not blank.

The Variation column is a binary value which is either Yes or No. If it is Yes, then it means that there is a variation (technical difference) between the National Sentence Text and the P/T Sentence Text and if it is No, it means that both the sentence texts mean the same (or editorial difference).

3 Preprocessing

3.1 Variations dataset

We begin by preprocessing the 2015 variations dataset. There are two groups of files based on the column names and the number of columns. The column names between the groups differ based on the strings 'Matched' and '?' present in a few of them. So, we remove them from the columns names using regex. Then, we add columns for 'Province/Territory' and 'Code Type'.

Next, we change one of the Province/Territory names 'PEI' to 'PE'. For the PE variations data, all of the sentences that are supposed to be in the 'P/T Sentence Text' fields, are in the 'P/T Article Title (FR)', except for one sentence. So, we move them back to the appropriate field.

Then, we utilize a function to preprocess the sentence texts and add a space in place of characters that follow the criteria, using regular expressions:

- characters not being rendered
- alphanumeric bullets at the beginning of the sentence (eg: '1)', 'a)')
- newline characters
- entire sentence being '-' or '_'
- specific characters not being rendered (eg: '\x93')
- alphanumeric bullets in the middle of a sentence, that are preceded and followed by a space (eg: ' a) ')
- question marks
- sentence number at the beginning of a few sentences (eg: '1.1.1.1.')
- two or more consecutive spaces
- trailing and leading spaces

Division B contains the majority of the technical provisions, and most of the sentence texts, variations, and other fields are complete in Division B of each Code book. Therefore, we separate Division B from the variations and proceed with it. We then populate the 'Difference Type' fields, whenever they are missing, based on the following criteria:

- 'Variation' == 'Yes'
  - 'National Sentence Text' is not blank and 'P/T Sentence Text' is blank, then 'Difference Type' is 'National Only'.
  - 'National Sentence Text' is blank and 'P/T Sentence Text' is not blank, then 'Difference Type' is 'P/T Only'.
- 'Variation' == 'No'

  - Both ‘National Sentence Text’ and the ‘P/T Sentence Text’ are not blank, then the ‘Difference Type’ is ‘Common Sentence’.

In the data, there exist rows where there are missing sentences even though the Difference Type is present. So, we remove rows that match the following criteria based on each Difference Type:

- Common Sentence: National Sentence Text or P/T Sentence Text or both are missing.
- National Only: National Sentence Text is missing.
- P/T Only: P/T Sentence Text is missing.

After this, we save the cleaned up Division B of the 2015 variations dataset. Them, we further split the cleaned up 2015 variations dataset based on the Province/Territory and save individual P/T data as csv files. We will work with seven P/Ts: Alberta, British Columbia, Nova Scotia, Nunavut, Ontario, Prince Edward Island, and Saskatchewan. We will not be working with Newfoundland and Labrador, since it the sentence texts are not present in the full excel files, and also Quebec for now, because it is written in French.

3.2 Full sentences

We read the National and P/T full sentence text data into data frames, preprocess the sentences using the same preprocessing function as above, isolate Division B from both of them, and remove any rows with missing sentence texts.

4 Creating train and evaluation sets

After cleaning both the 2015 variations dataset and the full sentences data, we split the 2015 variations dataset into train and evaluation sets. Since the objective is to compare sentences in each of the P/T Code books to sentences in the National Code book, we will split the variations dataset into train and evaluation sets based on the ‘National Sentence Text’ field. So, we have to make sure that there is no data leakage between the two sets, which means that there are no common National sentence texts between them.

To achieve this, we first split the National sentence texts in the full data into train and test sentences. A direct match between the sentences in the full data and the sentences in the variations data, even after preprocessing using regex, would not work because of differences in the way the same exact sentences are formatted in the two files. For example, consider these sentences:

- National Sentence Number: 9.15.3.4.(3)
  - Sentence in Full file: ‘Where a foundation rests on gravel, sand or silt in which the water table level is less than the width of the footings below the bearing surface, the footing width for walls shall be not less than twice the width required by Sentences 9.15.3.4.(1)and 9.15.3.4.(2), and Articles 9.15.3.5.and 9.15.3.6., and the footing area for columns shall be not less than twice the area required by Sentences 9.15.3.4.(1)and 9.15.3.4.(2) and Article 9.15.3.7.’
  - Sentence in Variations file: ‘Where a foundation rests on gravel, sand or silt in which the water table level is less than the width of the footings below the bearing surface, the footing width for walls shall be not less than twice the width required by Sentences 9.15.3.4.(1) and 9.15.3.4.(2), and Articles 9.15.3.5. and 9.15.3.6., and the footing area for columns shall be not less than twice the area required by Sentences 9.15.3.4.(1) and 9.15.3.4.(2) and Article 9.15.3.7.’

The only differences between the two sentences is that there are no spaces between certain characters and words in the sentence in the full file but the one in the variations file looks almost perfect with spaces wherever necessary. There are many such inconsistencies that need to be handled.

Since, we are trying to use the National and P/T full sentences to create vectors/embeddings, get the alignments, and then evaluate on the evaluation set (obtained from the variations dataset), we need to make sure that the train and evaluation sets contain only those sentences that are in the full sentences files. For this, we need to compare the sentences in the full files and the sentences in the variations files and include only those that match. So, we experiment with two approaches.

4.1 Computing Sentence Similarity Scores

Both the approaches involve computing a simple similarity score between sentences calculated as follows:

- Remove leading and trailing spaces from both sentences and split them based on spaces.
- Count the number of common words between them.
- Get the maximum of the lengths of both sentences.
- Compute the similarity score between the two sentences using the equation:

  $$Similarity\ Score = \frac{CommonWords}{MaxLength}$$

  where,
  CommonWords = number of common words between the two sentences
  MaxLength = max((length of sentence 1), (length of sentence 2))

Before proceeding to use any of the approaches, we isolate the not N/A sentence text fields in the full data and the not N/A sentence text fields in the variations dataset, for any one type (National or P/T). Then, we compute the Sentence Similarity Scores between each sentence in the full and each sentence in the variations. After this, we add the sentence in full, sentence in variations, and their sentence similarity score to a data frame. Finally, we arrange the rows based on the descending order of their scores.

4.1.1 Approach 1: Choosing a threshold

The first approach involves choosing a certain threshold for the Sentence Similarity Score manually and then including only those sentences with a score greater than or equal to that of the threshold. However, this approach has certain drawbacks:

- Some sentences are shorter in length compared to the others which makes the slightest difference between sentences in full and variations lower the score by a lot. For example, consider these National sentences taken from the variations dataset and the full data:
  - Except as provided in Articles 9.10.21.2.to 9.10.21.9., construction camps shall conform to Subsections 9.10.1.to 9.10.20.
  - Except as provided in Articles 9.10.21.2. to 9.10.21.9., construction camps shall conform to Subsections 9.10.1. to 9.10.20.
  - Similarity: 76.47%
- There are few sentences in the variations dataset that have the same structure but differ by just two or three words. this would mean that the same sentence in one file would be matched with multiple sentences in the other file with very high scores. This would make it difficult to pinpoint one unique match for a sentence in full data. For example, consider these sentences taken from the AB variations dataset:
  - A building classified as Group F, Division 3 is permitted to conform to Sentence 3.2.2.83.(2) provided it is not more than 4 storeys in building height, and it has a building area not more than the value in Table 3.2.2.83.
  - A building classified as Group F, Division 3 is permitted to conform to Sentence 3.2.2.83.(2) provided it is not more than 3 storeys in building height, and it has a building area not more than the value in Table 3.2.2.83.
  - Similarity: 97.5%

  The same sentence would be matched again with another sentence:
  - A building classified as Group A, Division 3 is permitted to conform to Sentence 3.2.2.30.(2) provided it is not more than 2 storeys in building height, and it has a building area not more than the value in Table 3.2.2.30.
  - A building classified as Group F, Division 3 is permitted to conform to Sentence 3.2.2.83.(2) provided it is not more than 3 storeys in building height, and it has a building area not more than the value in Table 3.2.2.83.
  - Similarity: 90%
- There are also cases where the same sentence text in one file has extra information compared to the one in the other file even though they are supposed to be the same sentence.
- Some sentence texts contains equations whose characters are not being rendered properly in one file. This causes the two sentences to have very low similarity score. For example, consider these sentences:

- The basic roof snow load factor, Cb, shall be determined as follows: Cb=0.8for lc 70C2w,and Cb=1Cw1-1-0.8Cwexp-lcC2w-70100forlc >70C2wwhere lc= characteristic length of the upper or lower roof, defined as 2w w2/l, in m, w= smaller plan dimension of the roof, in m, and l= larger plan dimension of the roof, in m, or conform to Table 4.1.6.2.-B, using linear interpolation for intermediate values of lcC2w. (See Note A-4.1.6.2.(2).)
- The basic roof snow load factor, Cb, shall be determined as follows: (i) Cb = 0.8 for for lc<= (70/C2w), and (b) 1/Cw [1 - (1 - 0.8Cw) exp (- (lcC2w-70)/100)] for for lc> (70/C2w) where, lc = characteristic length of the upper or lower roof, defined as 2w-wÂ²/l, in m, w = smaller plan dimension of the roof, in m, and l =larger plan dimension of the roof, in m, or conform to Table 4.1.6.2.-B., using linear interpolation for intermediate values of lcC2w
- Similarity: 61.90%

Tightening the threshold would mean that we are leaving out too many sentences that might be a match and relaxing the threshold would cause including sentences that are not exactly the same. Also, like mentioned earlier, since one sentence might be matched with multiple sentences in the other file, it would become a challenge, to retrieve sentences matched, when trying to evaluate the model later on. For these reasons, there is no fixed threshold we could choose for choosing the matching sentences.

4.1.2 Approach 2: Filtering the data frame twice

Approach 2 involves filtering the similarity score data frame twice to ensure that each sentence in the full data has the best possible match with at most one sentence in the variations dataset. Using the similarity score data frame whose rows have been arranged based on the descending order of the scores, we follow the steps below:

1. Initialize a list, to keep track of sentence texts in one file, and a new data frame, to add the filtered rows.
2. Iterate through the similarity score data frame, while keeping track of the sentence texts in 'Sentence in Full' column.
3. If the sentence text does not exist in the list, add it to the list, and also add the entire row to the new data frame. If it does exist, ignore the row and move on to the next row. Once we have iterated through the entire similarity data frame, we would have found the best matches, with the highest similarity scores, for the sentence texts in the full data.
4. After filtering out like above, we would still have duplicate sentence texts in the 'Sentence in Variations' column (the number of sentences in full data are greater than the number of sentences in variations). So, we filter the filtered data frame from above again using the same method but keeping track of the sentences in 'Sentence in Variations' column this time.

The final data frame would have best possible matches between the sentences in the full data and the sentences in the variations dataset. We can then use the sentences in the 'Sentences in Variations' column to make sure we are including only those in the train and evaluation sets.

4.2 Train and evaluation splits

We use Approach 2 to get the perfect matches between sentence texts in full data and the sentence texts in variations data for both National and P/T Codes. Then, we follow the steps below:

1. The variations dataset is split into train and evaluation sets based on the unique national train sentences and national test sentences we have from the full sentence data, if the National Sentence Text is also present in the National Similarity score data frame. We also check and include only those rows where 'P/T Sentence Text' is either in the P/T Similarity score data frame or it is N/A (cases where the Difference Type is National Only). Let us call them *train* and *test*.
2. After this, we would still need to include rows which have empty 'National Sentence Text' fields (for Difference Type as P/T Only). So, we would isolate only those rows from the variations dataset where the sentences in the 'P/T Sentence Text' field are

present in the P/T Similarity score data frame and we will call this data frame *empty*.

3. We finally split the *empty* dataset into train and test subsets, and further concatenate them to *train* and *test*, from above, to get the final sets based on a 80/20 split.

Using the method above, we create the train and evaluation sets from the 2015 variations dataset for each Province/Territory and Table 1 displays the sizes of the train and test sets after splitting using Approach 2.

After getting the train and evaluation sets, we notice that there are some rows that are mislabelled as 'P/T Only' or 'National Only' even though both the National and P/T Sentence Texts are present and the 'Variation' == 'No'. We change the 'Difference Type' to 'Common Sentence' for rows that match this criteria in BC train, BC test, and ON train sets.

Province/Territory	**Train set**	**Evaluation set**
AB	777	194
BC	742	186
NS	58	14
NU	6	2
ON	6347	1587
PE	33	8
SK	35	9

Table 1: Training and Evaluation Set Counts by Province/Territory, created using Approach 2

5 Experiments

We now conduct our experiments on different models to get the alignments of 'National Sentence Text' and the 'P/T Sentence Text'. The main pipeline for our experiments is described as below

1. Load the National and P/T full sentence data into two data frames.

2. If using baselines, we preprocess the sentence texts using the PunktSentenceTokenizer() from nltk followed by adding spaces after a closing parenthesis or a period, if they are not followed by a character.

3. For baselines (traditional NLP methods), we create a corpus using all the sentences, and then covert the full national sentences and the full P/T sentences into vectors based on the type of baseline model. For transformer-based models, we get the vector embeddings using the full National and P/T sentences as input.

4. Using the vectors/embeddings, we compute the cosine similarity scores for each sentence in National with each sentence in P/T and get the cosine similarity matrix.

5. We now have a bipartite weighted graph with the National sentences and the P/T sentences as the two groups with the cosine similarity scores as their weighted edges. Since, the higher the cosine similarity score, the more closer the sentences are in space, we use the maximum weight matching algorithm and get the indices, for determining common sentences. A matching or an alignment means that the 'National Sentence Text' and the 'P/T Sentence Text' have been aligned together as Common Sentence.

6. The number of National sentences is not necessarily equal to the number of P/T sentences. So, sentences at the indices that are not returned would be unmatched or unaligned, which means that those sentences would either be 'National Only' or 'P/T Only'. We add all the aligned and unaligned sentences, along with their cosine similarity scores to a new data frame (*Alignment*) based on the indices. If the sentence is unaligned, the cosine similarity score would simply be 0.

7. The maximum weight matching algorithm tends to over align sentences, meaning that there are going to be a lot of sentences that would have been aligned together with very low similarity scores. We need to break up those alignments into 'National Only' and 'P/T Only'. In order to do this, we must choose a certain threshold so that, all the alignments with a score below that threshold will be broken up and all those above the threshold will be retained. The approach below describes a way to solve this:

   (a) For a range of thresholds from 0 to 1.0, in the increments of 0.01, use each threshold on the *Alignment* data frame, keep the alignments for rows equal to or above the threshold, and split up the alignments for those below.

(b) Calculate the Alignment Error Rate on the train set as follows:

$$AER = 1 - \left(\frac{2 \times Correct}{Predicted + Actual}\right)$$

where,
Correct = number of correct alignments
Predicted = number of predicted alignments
Actual = number of actual alignments

(c) Plot a graph of the Alignment Error Rates vs. Thresholds for the calculated AER and choose the threshold corresponding to the lowest AER. The graphs would look like Figures 1a to 7c.

(d) Now, the optimal threshold would be the threshold corresponding to the lowest point in the graph.

(e) Use the optimal threshold, from above, on the *Alignment* data frame and get the final Alignment Error Rate on the test set.

5.1 Models

We run our experiments on three baseline models as well as several transformer-based models described below.

5.1.1 Baselines

- Bag-of-Words:
  - This model converts a sentence into a vector based on a simple word count.
- TF-IDF weighted Bag-of-Words:
  - This model creates a vector based on the words counts weighted on the Term Frequency - Inverse Document Frequency.
  - Term Frequency is based on a concept that says that the higher the frequency of a word in a sentence, the lower is its importance. For example, most common words like the, and, is, etc. would have low importance.
  - Inverse Document Frequency says that the lower the frequency of a word across the entire corpus, the higher is its importance. For example, terms related to the construction domain like building, sprinklers, corridors, etc. would have very high importance.
- 1-Hot Encoding:
  - This model creates a vector based on a binary encoding. If the word from the vocabulary is present in the sentence, the index in the vector is set to 1, else 0.

5.1.2 Transformer-based

We utilize six pre-trained models to convert the complete National and P/T Sentences into vector embeddings. The models were selected from the MTEB leaderboard from HuggingFace [1], for the task of Sentence Text Similarity for English, based on the highest average Spearman correlation and practical model sizes. The models used along with the descriptions of why they are suitable in our case are as follows:

- mxbai-embed-large-v1 [2]
  - # of parameters: 335 million
  - Memory usage: 1.25 GB
  - Training and fine-tuning:
    - Trained on 700 million pairs using contrastive training.
    - Fine-tuned on 300 million triplets with Angle loss function that mitigates the risk of vanishing gradients.
  - This model has the ability to generalize well across various several domains, tasks, and text lengths. Subtle differences in text are pointed out because of the Angle loss function.
- multilingual-e5-large-instruct [3]
  - # of parameters: 560 million
  - Memory usage: 2.09 GB
  - Training and fine-tuning:
    - Trained on 1 billion multilingual text pairs from numerous sources.
    - Fine-tuned on a variety of datasets: question-answering, inference, fact extraction verification, information retrieval across different languages.
  - Being trained on various types of datasets, this model has been known to outperform English-only models on various tasks.
- GIST-large-Embedding-v0 [4]
  - # of parameters: 335 million
  - Memory usage: 1.25 GB

  - Training and fine-tuning:
    * Trained on MEDI. that contains 1,450,000 triplets of query, positive, and negative examples, and 11 MTEB classification datasets (with training splits not used during evaluation).
    * The model used existing models to guide during fine-tuning.
  - This model used a method to reduce noise and bias along with addressing potential data quality issues.

- UAE-Large-V1 [2]
  - # of parameters: 335 million
  - Memory usage: 1.25 GB
  - Training and fine-tuning:
    * This model used the same training methodology as the mxbai model from above

- GIST-Embedding-v0 [4]
  - # of parameters: 109 million
  - Memory usage: 0.41 GB
  - Training and fine-tuning:
    * This model used the same training methodology as the GIST model from above.

- GIST-small-Embedding-v0 [4]
  - # of parameters: 33 million
  - Memory usage: 0.12 GB
  - Training and fine-tuning:
    * This model used the same training methodology as the GIST models from above.

5.2 Fine-tuned

We also fine-tune the GIST-Embedding-v0 model with contrastive training because it seems to provide the lowest Alignment Error Rates across various Province/Territories. Contrastive training leverages positive and negative sentence pairs to teach the model how to differentiate between similar and dissimilar sentences. This process helps the model push dissimilar sentences further apart and pull similar sentences closer together in the embedding space. We create the positive and negative pairs for contrastive training, using our dataset, as follows:

- Positive pairs: 'National Sentence Text' and 'P/T Sentence Text' where 'Difference Type' is 'Common Sentence'.

- Negative pairs: 'National Sentence Text' with 'Difference Type' as 'National Only' combined with 'P/T Sentence Text' with 'Difference Type' as 'P/T Only'.

6 Results

Tables 2 and 3 present the calculated Alignment Error Rates on the test sets for each Province/Territory and the corresponding thresholds used. The following observations can be made based on the results:

- There is not a significant improvement in using the transformer-based models for creating vector embeddings compared to using the baselines.

- The AERs for BC and ON seem significantly higher, for their test sizes, compared to the other P/Ts because, there exist some data quality issues meaning that a lot of sentences in one Code book were labelled as National Only or P/T Only, even though they had a match with a sentence in the other Code book.

- The AER for SK is high for a few models because of the way we are choosing the threshold. We got the final AER, using the first best threshold. However, there are a lot of cases for different P/Ts, where the same lowest Alignment Error Rate occurs more than once for multiple thresholds on the train set. Consider the following example:
  - In Figure 7c, the thresholds from 0.78 till 0.99 seem to give the same Alignment Error Rate, which is the lowest. We chose 0.78 as our threshold. Trying to modify the approach to choose 0.99 might be helpful since, it would mean that there would be less sentences being aligned together as Common Sentence.

  It might be worth trying to use the last possible threshold corresponding to the lowest Alignment Error Rate on the train set.

7 Extracting 2020 data

The 2020 XML files have the following hierarchical structure of <part><section><subsect><article><sentence>. We use this to extract the data from these XML files.

So far, we have managed to extract all the data for AB fire, AB building, and National building codes. But, the numbering is yet to be done for the sections, subsections, and the sentences. For now, these fields have IDs in them and they follow a pattern for numbering. Also, there are references to certain articles, sentences, tables, and notes. And the IDs are informative in the sense that if it referring to a note, the ID would start with an 'en', if it is referring to a sentence, it would start with an 'es' and so on.

8 Future Work

- Find the Variation: Yes vs. No based on the alignments we already have and using F-1 score as the evaluation metric on them, as we want a balance between the Precision and Recall.
- Find these variations for Quebec, which is in French.
- Clean the sentences in the 2020 data and number them.
- Use the 2020 dataset to find the variations and label them accordingly.

9 Integration/Deployment

The work that has been done so far for this project has all been implemented using python. All of the python scripts were run locally using the files downloaded on to the laptop. The output would be integrated into an in-progress tool called the Codes Comparison Tool, that renders the difference between a National Sentence and a P/T Sentence with an HTML format for use by members of the Construction department.

10 Business Case

Everything was run locally and all the models used are open source. The solution would reduce a lot of manual work and time, that would typically require people comparing each line in one Code Book with each line in another Code book.

References

[1] Muennighoff, Niklas and Tazi, Nouamane and Magne, Loïc and Reimers, Nils, *MTEB: Massive Text Embedding Benchmark*, arXiv preprint arXiv:2210.07316, 2022. https://arxiv.org/abs/2210.07316

[2] Li, Xianming and Li, Jing, *AnglE-optimized Text Embeddings*, arXiv preprint arXiv:2309.12871, 2024. https://arxiv.org/abs/2309.12871

[3] Wang, Liang and Yang, Nan and Huang, Xiaolong and Yang, Linjun and Majumder, Rangan and Wei, Furu, *Multilingual E5 Text Embeddings: A Technical Report*, arXiv preprint arXiv:2402.05672, 2024. https://arxiv.org/abs/2402.05672

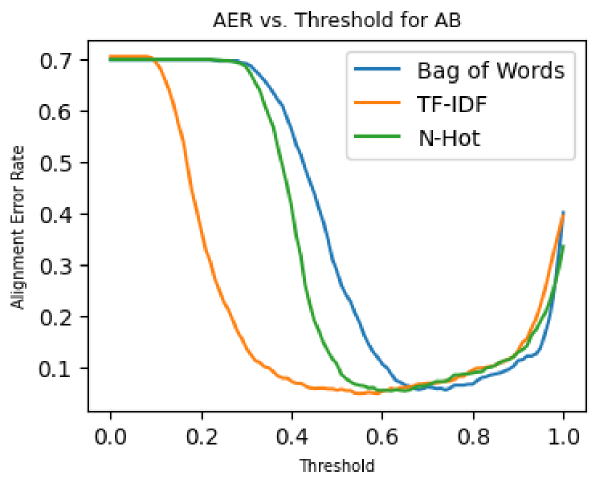
[4] Solatorio, Aivin V., *GISTEmbed: Guided In-sample Selection of Training Negatives for Text Embedding Fine-tuning*, arXiv preprint arXiv:2402.16829, 2024. https://arxiv.org/abs/2402.16829

A Appendix

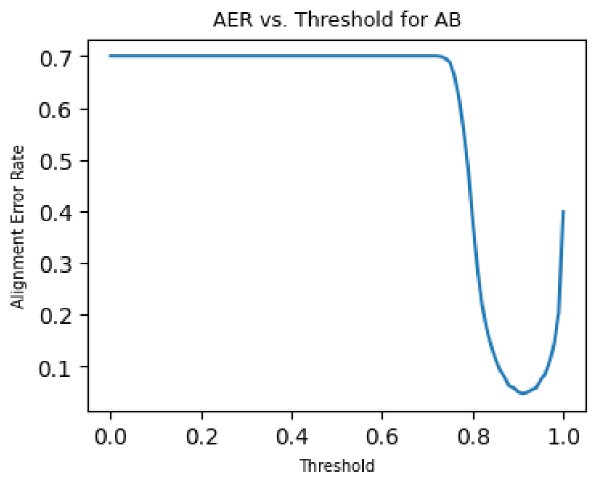
This appendix has the plots for the Alignment Error Rates vs. Thresholds on the train set for each Province/Territory. Each figure represents the plots for one P/T and contains three sub-figures:

- First: The AER vs. Threshold on the train set for the three baselines: Bag of Words, TF-IDF weighted Bag of Words, and 1-Hot Encoding (labelled as N-Hot in the plot).
- Second: The AER vs. Thresholds on the train set for the pre-trained model: GIST-small-Embedding-v0
- Third: The AER vs. Threshold on the train set for the fine-tuned GIST-Embedding-v0 model.

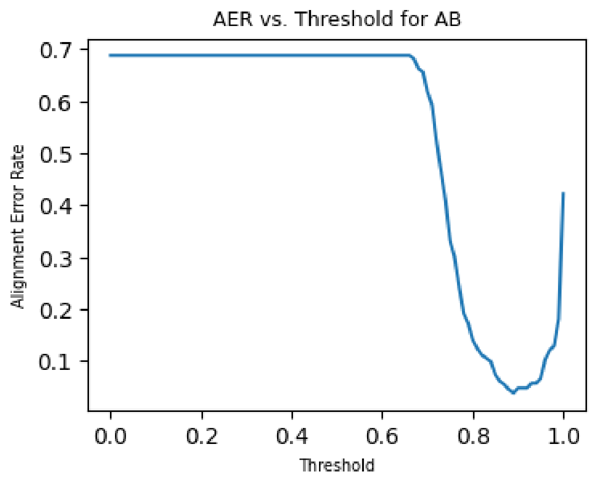
It also includes two sections of the table that display the alignment error rate at the threshold, based on the optimal threshold used for each model type across all territories.



(a) Baselines

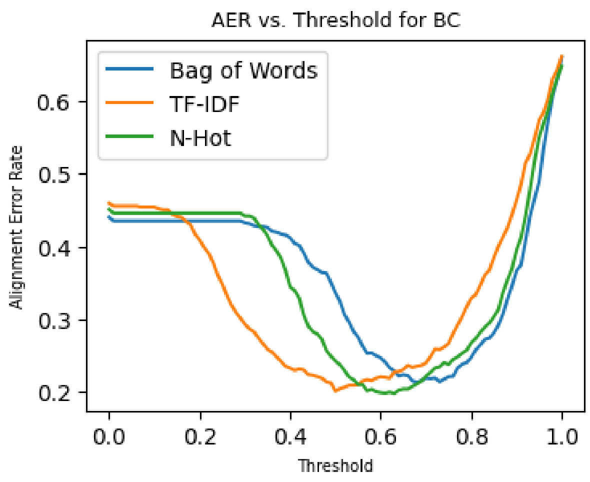


(b) Transformer-based

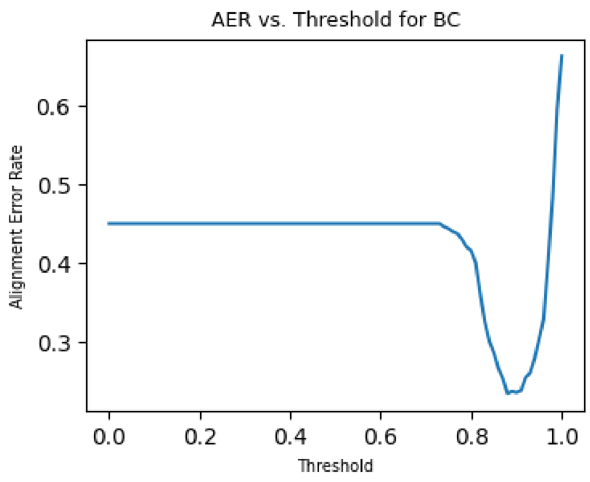


(c) Fine-tuned

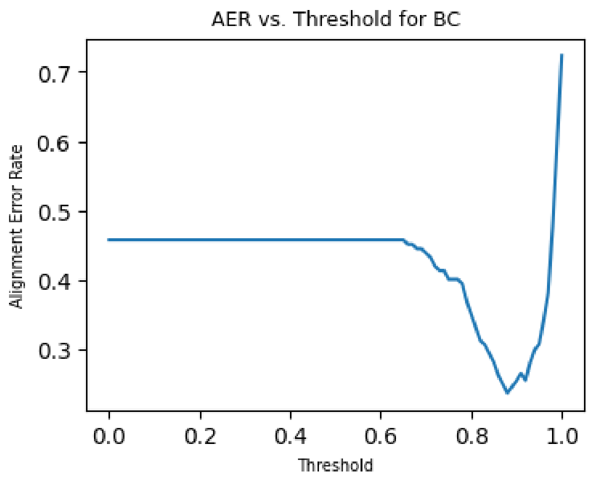
Figure 1: AER vs. Threshold for AB using different methods



(a) Baselines

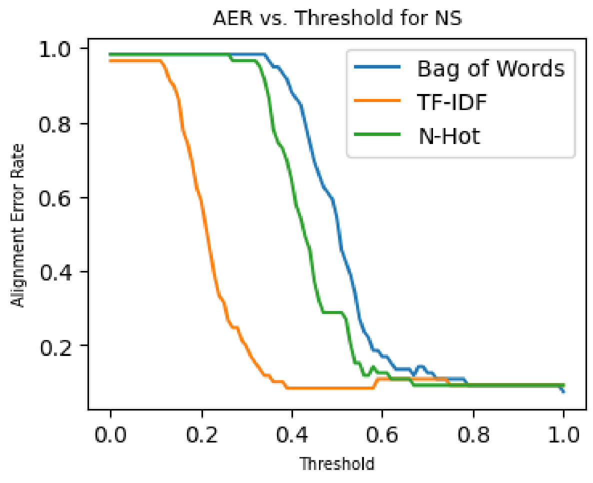


(b) Transformer-based

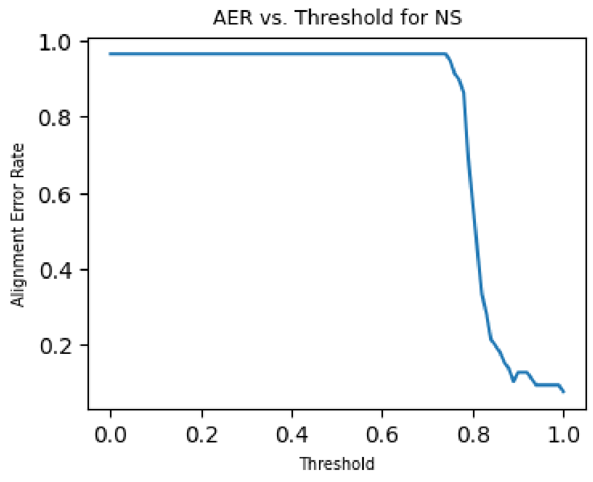


(c) Fine-tuned

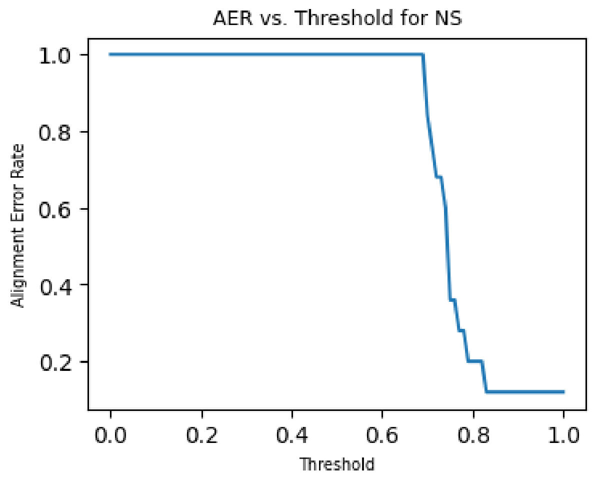
Figure 2: AER vs. Threshold for BC using different methods



(a) Baselines

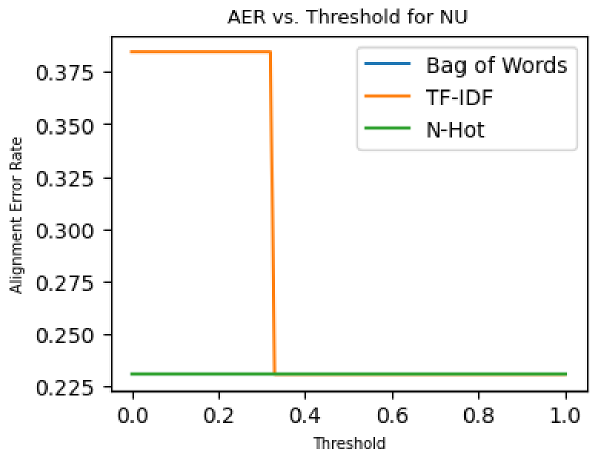


(b) Transformer-based

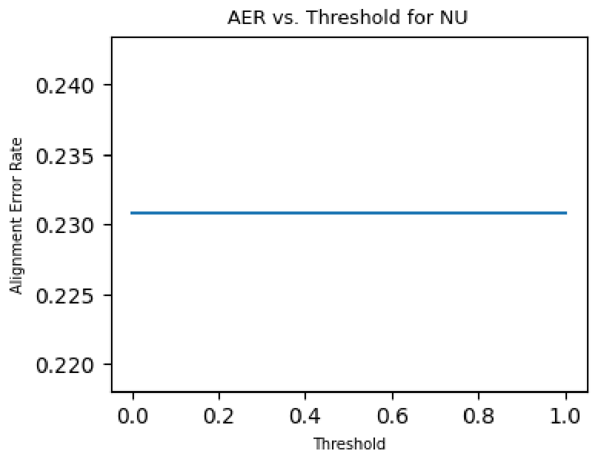


(c) Fine-tuned

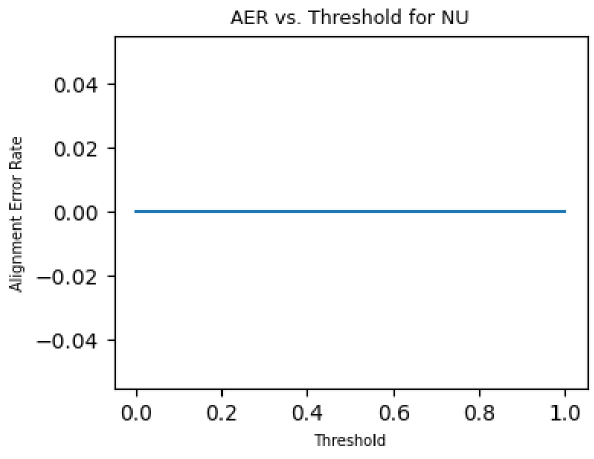
Figure 3: AER vs. Threshold for NS using different methods



(a) Baselines

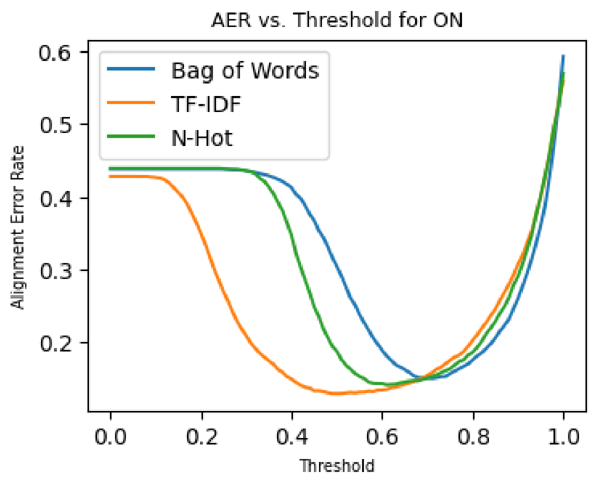


(b) Transformer-based

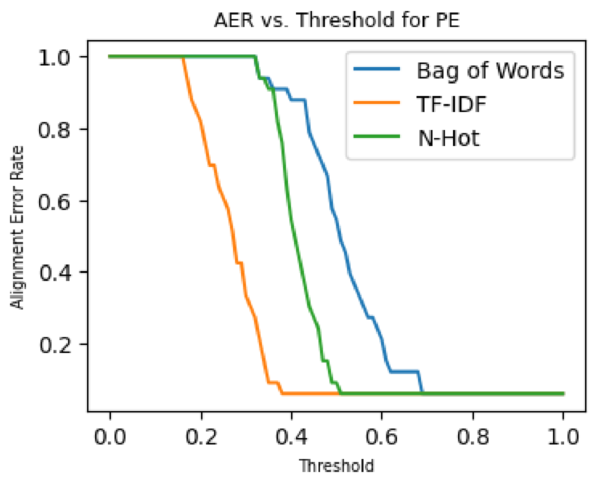


(c) Fine-tuned

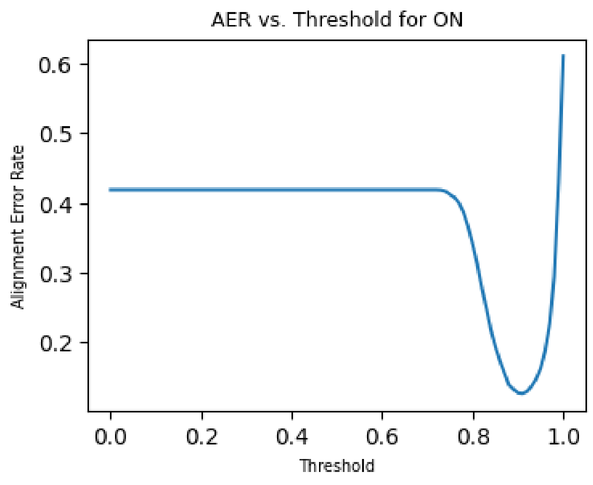
Figure 4: AER vs. Threshold for NU using different methods



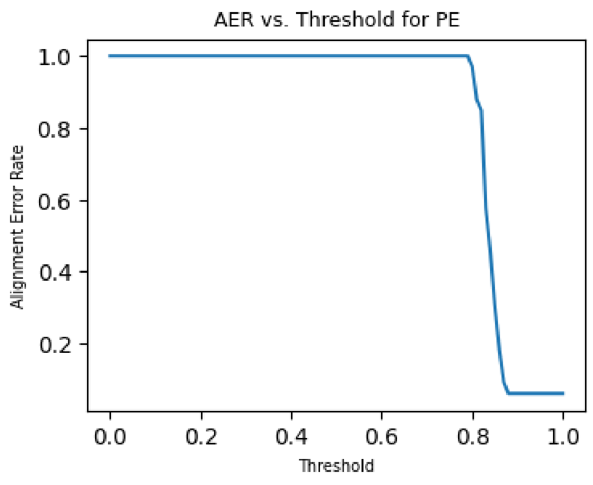
(a) Baselines



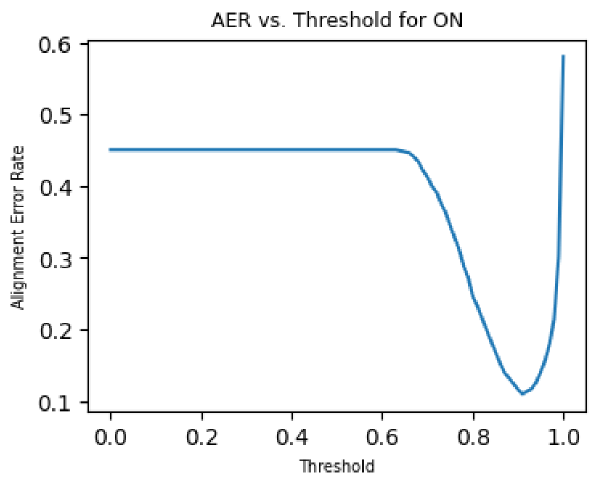
(a) Baselines



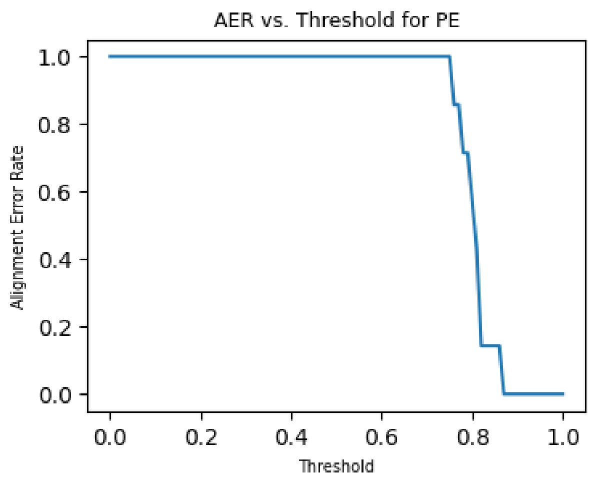
(b) Transformer-based



(b) Transformer-based



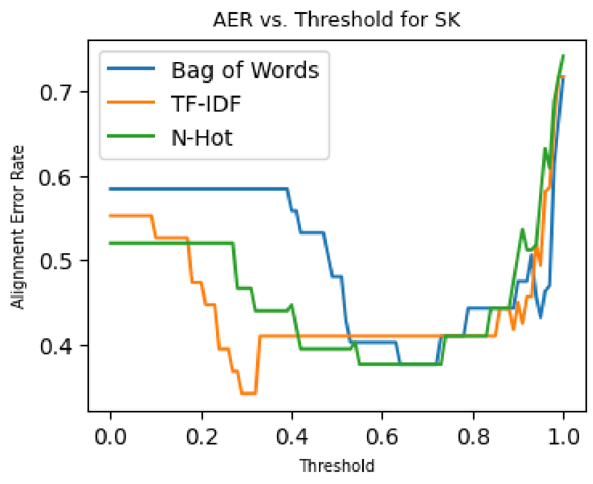
(c) Fine-tuned



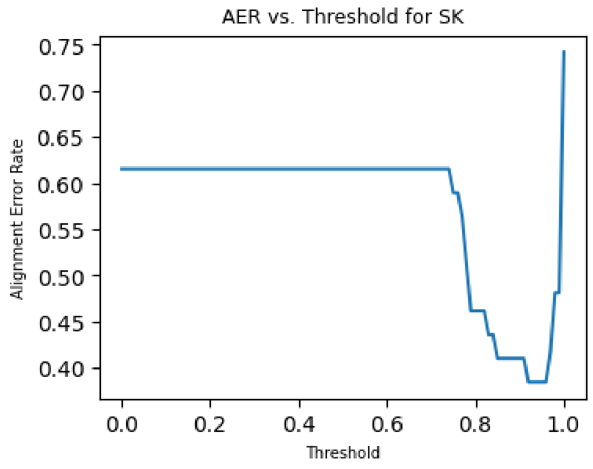
(c) Fine-tuned

Figure 5: AER vs. Threshold for ON using different methods

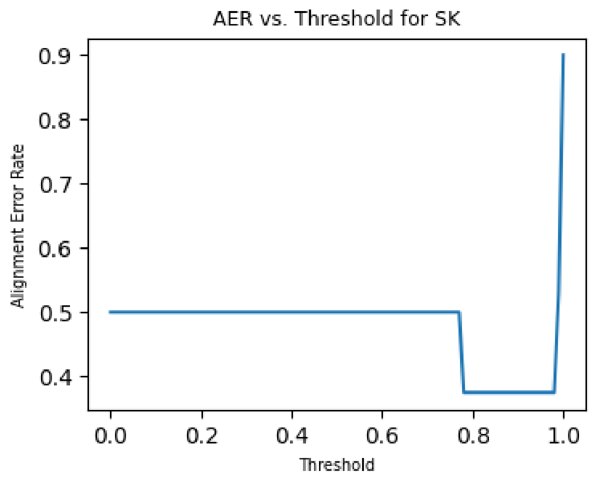
Figure 6: AER vs. Threshold for PE using different methods



(a) Baselines



(b) Transformer-based



(c) Fine-tuned

Figure 7: AER vs. Threshold for SK using different methods

P/T Name	Model Type	Model Name	Best Threshold	Alignment Error Rate
AB	Baselines	Bag-of-Words	0.74	0.0758
		TF-IDF	**0.59**	**0.0633**
		N-Hot Encoding	0.65	0.0657
	Transformer-based	mxbai-embed-large-v1	0.88	0.0730
		multilingual-e5-large-instruct	0.95	0.0657
		GIST-large-Embedding-v0	0.88	0.0730
		UAE-Large-V1	0.87	0.0804
		GIST-Embedding-v0	**0.90**	**0.0357**
		GIST-small-Embedding-v0	0.91	0.0582
	Fine-tuned	GIST-Embedding-v0	0.89	0.0357
BC	Baselines	Bag-of-Words	0.68	0.2525
		TF-IDF	0.50	0.2205
		N-Hot Encoding	**0.63**	**0.2031**
	Transformer-based	mxbai-embed-large-v1	0.85	0.2205
		multilingual-e5-large-instruct	**0.93**	**0.2093**
		GIST-large-Embedding-v0	0.81	0.2185
		UAE-Large-V1	0.84	0.2174
		GIST-Embedding-v0	0.87	0.2165
		GIST-small-Embedding-v0	0.88	0.2113
	Fine-tuned	GIST-Embedding-v0	0.88	0.2134
NS	Baselines	Bag-of-Words	1.00	0.1034
		TF-IDF	**0.39**	**0.0000**
		N-Hot Encoding	0.67	0.1034
	Transformer-based	mxbai-embed-large-v1	1.0	0.1034
		multilingual-e5-large-instruct	1.0	0.1034
		GIST-large-Embedding-v0	1.0	0.1034
		UAE-Large-V1	1.0	0.1034
		GIST-Embedding-v0	1.0	0.1034
		GIST-small-Embedding-v0	1.0	0.1034
	Fine-tuned	**GIST-Embedding-v0**	**0.83**	**0.0000**
NU	Baselines	Bag-of-Words	0.00	0.0000
		TF-IDF	0.33	0.0000
		N-Hot Encoding	0.00	0.0000
	Transformer-based	mxbai-embed-large-v1	0.0	0.0000
		multilingual-e5-large-instruct	0.0	0.0000
		GIST-large-Embedding-v0	0.0	0.0000
		UAE-Large-V1	0.0	0.0000
		GIST-Embedding-v0	0.0	0.0000
		GIST-small-Embedding-v0	0.00	0.0000
	Fine-tuned	GIST-Embedding-v0	0.0	0.0000

Table 2: Alignment Error Rates of Different Models for Each P/T (Part 1)

P/T Name	Model Type	Model Name	Best Threshold	Alignment Error Rate
ON	Baselines	Bag-of-Words	0.71	0.1626
		TF-IDF	0.50	0.1510
		N-Hot Encoding	**0.61**	**0.1487**
	Transformer-based	mxbai-embed-large-v1	0.86	0.1413
		multilingual-e5-large-instruct	0.94	0.1496
		GIST-large-Embedding-v0	0.83	0.1408
		UAE-Large-V1	0.84	0.1411
		GIST-Embedding-v0	**0.9**	**0.1360**
		GIST-small-Embedding-v0	0.91	0.1397
	Fine-tuned	GIST-Embedding-v0	0.91	0.1463
PE	Baselines	**Bag-of-Words**	**0.69**	**0.0000**
		TF-IDF	0.38	0.1250
		N-Hot Encoding	**0.51**	**0.0000**
	Transformer-based	mxbai-embed-large-v1	0.79	0.0000
		multilingual-e5-large-instruct	0.93	0.0000
		GIST-large-Embedding-v0	0.78	0.1250
		UAE-Large-V1	1.0	0.0000
		GIST-Embedding-v0	0.89	0.0000
		GIST-small-Embedding-v0	0.88	0.0000
	Fine-tuned	GIST-Embedding-v0	0.87	0.0000
SK	Baselines	**Bag-of-Words**	**0.64**	**0.1579**
		TF-IDF	0.29	0.2222
		N-Hot Encoding	0.55	0.2632
	Transformer-based	mxbai-embed-large-v1	0.74	0.3333
		multilingual-e5-large-instruct	0.91	0.1111
		GIST-large-Embedding-v0	**0.91**	**0.0000**
		UAE-Large-V1	0.72	0.3333
		GIST-Embedding-v0	0.92	0.1579
		GIST-small-Embedding-v0	**0.92**	**0.0000**
	Fine-tuned	GIST-Embedding-v0	0.78	0.4737

Table 3: Alignment Error Rates of Different Models for Each P/T (Part 2)