Relation Extraction on MIMIC-III Data

Redoan Rahman
School of Information
The University of Texas at Austin
Austin, TX
redoan.rahman@utexas.edu

Dhanny Indrakusuma
School of Information
The University of Texas at Austin
Austin, TX
dhannywi@utexas.edu

Patrick Sui
School of Information
The University of Texas at Austin
Austin, TX
peiqisui@utexas.edu

ABSTRACT

The research project focuses on exploring different possibilities to build a successful model that can classify the type of relation type available in a sentence of medical text. In our research, we utilized a variety of feature extraction methods such as TF-IDF, Bag-of-Words, Word2Vec, Spacy, BERT, and Sentence-BERT. For the classification task, we implemented models such as Decision Tree, Random Forest, Long Short Term Memory model et cetera. We found that in our smaller datasets, simpler models do surprisingly better than the complex model. We also found that SentenceBERT provides excellent representation which improved the results for almost all of the classification models. We will use our findings to further advance the research.

Keywords

SHAC, Relation Extraction, BERT, LSTM, Pre-trained BERT, SentenceBERT

1. INTRODUCTION

As a vital part of evidence-based care, social determinants of health (SDOH) can play a crucial role in the decision-making process of health professionals. Information on the history of substance abuse, for instance, could inform physicians on their prescription choices, while the employment and living status could give caregivers a fuller picture of the possible latent impacts on health.[1] The inclusion of SDOH in clinical decision-making helps to humanize medical procedures and allows the patients to feel that they are receiving customized care.[1]

Obtaining SDOH information, however, can be quite complicated. Structured data in publicly available databases like MIMIC-III only contain a fraction of possible SDOH to be retrieved. As far as the unstructured clinical notes are concerned, most social determinants are not explicitly spelled out or even directly mentioned, but implied or glossed over in curt or fragmented phrasing.[1] This requires them to be specifically annotated from the unstructured texts and notes, in order to reduce ambiguity and accommodate for faster records searches and retrievals. Doing this data entry task manually is very costly, and the sheer volume of the workload calls for AI solutions to address this as a relation extraction problem between two textual entities.

2. RELATED WORKS

Event and relation-oriented text annotation has been a well-researched area for biomedical informatics. As early as 2012, Stenetorp et al. trained an NLP-assisted rapid annotation tool (BRAT) that uses POS-tagging and named entity recognition to achieve multicategory entity annotation for events. [2] BRAT has since become the baseline preprocessing method for attempts to extract more detailed and fine-grained SDOH information, including Gehrmann et al. [3], Wang et al. [4], and Yetisgen and Vanderwende [5]. The most up-to-date development in the area was from Lybarger et al. in 2021, whose contributions include the Social History Annotation Corpus (SHAC), a new active learning framework that involves BERT and bi-LSTM, and the extraction results that their model pulled from SHAC. [1]

This paper will build on the results of Lybarger et al. by attempting to turn the relation extraction problem into a classification one. Implementing relation extraction and all its underlying queries on new instances of unlabeled data is computationally expensive, and it is more efficient to pass the data into a classifier for a multiclass or binary prediction. Due to the scale of this project, we will only focus on relation types argument (i.e., status, duration, history, type, amount, etc) instead of the more detailed subtypes argument (i.e., employed, current, alone, etc.) or span examples.

For a more thorough literature review on the construction of SDOH corpora and their application in active learning, please consult Section 2 of Lybarger et al.

3. DATASET

The corpus that this paper utilizes is drawn from SHAC published by Lybarger et al. It contains publicly available discharge summaries drawn from MIMIC-III, alongside the social annotation results extracted by their active learning methods. The higher-level event types, like amount, status, or type, are collected with the BRAT rapid annotation tool.

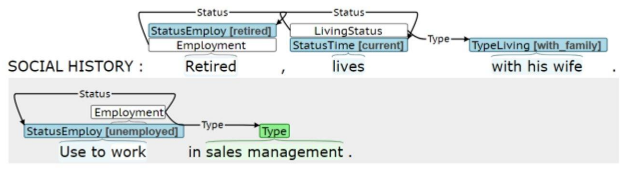


Fig 1. Example for BRAT annotation of living status [2]

The dataset consists of 3008 text and annotation files in total. Each text file contains notes describing certain information, and its corresponding annotation file contains the entities and relation types of each sentence in the text file. Each sentence directly corresponds to the representation of an event.

3.1 Data Preprocessing

Prior to feeding the data into our models, the Train and Dev annotated dataset was first processed using a custom algorithm developed specifically for this dataset. This is done to extract the sentence's relation type, the first and second entities (identified as T1 and T2) as well as their respective types, starting and ending positions. The pseudocode of the custom algorithm used to preprocess the dataset is shown in Fig 2.

The resulting dataframe is described below:

- **id**: The unique IDs are constructed by (name_of_the_file_without_extension+ number_of_sentence/event+ first_5_character_of_the_sentence). Except for identifying the source file of sentences, the ID also allows us to determine which sentences might share a contextual connection.
- **sentence:** The original sentence that was extracted from the file. Sentences were extracted using NLTK's tokenizer from the text file. The additional characters (whitespace and punctuations) were preserved to ensure that the position values of the entities and sentences in files could be calculated correctly.
- **T1:** The first entity of the event in the sentence.
- **T1_starting_position**: The starting position of the first entity in the sentence.

```
Input: unlabeled txt files of medical notes U, annotations A
Output: a preprocessed data frame tokenized into sentences and
    relation types dataset

SET textFiles TO name of all text file
SET annotationFiles TO name of all corresponding annotation file
INITITALIZE  dataFrame TO EMPTY_FRAMES
FOR currentTextFileName IN textFiles
              SET rawTextData TO text in currentTextFileName
   SET currentPrefix TO currentTextFileName without extension
      IF annotation file does not exist for currentTextFileName
         skip currentTextFileName
      ELSE
         SET currentAnnotationFile TO matching annotation file from
annotationFiles
              ENDIF
   SET rawAnnData TO text in currentAnnotationFile
   IF rawAnnData IS EMPTY
      skip currentTextFileName
   ENDIF
   FOR line_of_data in rawAnnData
      INITITALIZE triggers TO EMPTY_DICT
      INITITALIZE  events TO EMPTY_DICT
      INITITALIZE  values TO EMPTY_DICT
      IF line_of_data is about an entity
         INSERT start_pos, end_pos, the entity itself, and the type of entity
into triggers
      ELSEIF line_of_data is about an event
         INSERT the sequence of entities and the relation_type into events
      ENDIF
      SET sentenceInfoList TO []
      SET sentenceList TO all sentences in rawTextData
      FOR sentence IN sentenceList
         INSERT start_pos, end_pos, the sentence itself, and the number of
the sentence into sentenceInfoList

      FOR event in events
         FOR sentenceInfo in sentenceInfoList
            SET sentence TO sentenceInfo.sentence
            IF sentence corresponds to event
               SET eventSeq TO sequence of entities for the current    event
               SET t1EntityInfo TO 1st element of the sequence
               SET t1EntityId TO event.triggerId
               SET t1Entity TO triggers[t1EntityId]
               FOR entity in eventSeq starting from the 2nd position of the
sequence
                  SET t2EntityId TO entity.triggerId
                  SET t2Entity TO triggers[t1EntityId]
                     Insert sentence, relation_type from sentenceInfo,
                     T1, T1_starting_postion, T1_ending_position, T1_type
from t1Entity,
                     T1, T1_starting_postion, T1_ending_position, T1_type
from t2Entity
                     INTO dataFrame
                       ENDFOR
          ENDFOR
       ENDFOR
   ENDFOR
ENDFOR
CREATE dataset FROM dataFrame
END
```

Fig 2. Pseudocode of the data preprocessor

- **T1_ending_position**: The ending position of the first entity in the sentence.
- **T1_type:** The type value of the first entity (i.e., Alcohol, Drug, Employment, Living Status, Tobacco)

- **T2:** The second entity of the event in the sentence.
- **T2_starting_position**: The starting position of the second entity in the sentence.
- **T2_ending_position**: The ending position of the second entity in the sentence.
- **T2_type:** The type value of the second entity (i.e., Amount, Duration, Frequency, History, Method, StatusEmploy, StatusTime, Type, TypeLiving).
- **relation_type:** The general argument type of the relation between the entities of the sentence, extracted by BRAT. Possible values include Amount, Duration, Frequency, History, Method, Status, and Type. This is the label of our experiments, and we treat each individual relation_type as the target for classification.

4. METHODOLOGY

4.1 Workflow Summary

In our process, we preprocess the dataset into the appropriate format. Then we apply several feature extraction methods to them. We put information such as initial weight into the model. Then we get our results from the model and compare and analyze the success of different models. The process is visually demonstrated in Fig. 3

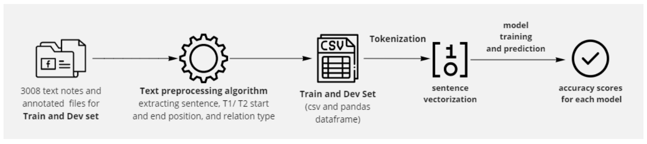

3008 text notes and annotated files for Train and Dev set
Text preprocessing algorithm extracting sentence, T1/ T2 start and end position, and relation type
Train and Dev Set (csv and pandas dataframe)
Tokenization
sentence vectorization
model training and prediction
accuracy scores for each model

Fig 3. Classification model workflow

4.2 Feature Extraction

We employed or attempted to employ several different feature extraction methods on the medical notes that would allow us to train our models. These feature extraction methods are explained below:

4.2.1 Bag-of-Words

Bag-of-Words is an approach to "preprocess the text by converting it into a bag of words, which keeps a count of the total occurrences of most frequently used words."[6] The resulting data is a vector representation of each word and its count in the given text.[6] In our model, the Bag of Words was created using scikit-learn's CountVectorizer to get the frequency of all words available in all sentences in the dataset.[6] Next, we used the frequency distribution of the words in a sentence as a vector representation of the sentence. The resulting list of vectors was used as our feature matrices and the binary representation of relation types was used as our target.

4.2.2 TF-IDF

Term Frequency Inverse Document Frequency (TF-IDF) is "a statistic that aims to better define how important a word is for a document, while also taking into account the relation to other documents from the same corpus".[7] TF-IDF looks at the number of times a word appears in a document as well as the number of times that exact word appears in other documents in the given corpus.[7] To generate TF-IDF scores for the sentences, we utilized scikit-learn's TfIdfVectorizer to get the frequency of all words available in every sentence in the dataset.[7] It computes the frequency distribution of the words in a sentence as a vector representation of the sentence.[7] The resulting list of vectors was used as our feature matrices and the binary representation of relation types was used as our target.

4.2.3 Word2Vec

Word2Vec model is part of gensim, a topic modeling library used to extract features from sentences to help figure out the main ideas in the sentences from a text. Within Word2Vec, we used a Phrase detector to identify unigram bigram and trigram features.
First, we converted the corpus using the bigram and trigram phrase detector and then transformed texts into sequences of 15 words.
The list of sequences was later used as our feature matrix. We converted all the words from the pre-processed corpus into embedding vectors using the pre-trained gensim model and the embeddings were then used to initialize the weights of the training model.

4.2.4 Word2Vec with spaCy

Next, we created a custom model that utilized Word2Vec and Spacy. According to spaCy's official documentation, "spaCy supports a number of transfer and multi-task learning workflows that can often help improve your pipeline's efficiency or accuracy".[8] For this custom model, we loaded the en_core_web_sm library of Spacy and converted all named entities into Spacy embedded representation.[8] The spaCy embedded corpus was then trained on the word2vec model and used to convert the corpus into a list of sequences of 50 words. The list of sequences was used as a feature matrix and the weights of the Word2vec model were used as initialization weights.

4.2.5 BERT

Bidirectional Encoder Representations from Transformers (BERT) was used to convert the corpus by adding appropriate word tokens (CLS, SEP) that are used by the

pre-trained BERT model to identify the start and end of sentences, and then broke the text into sequences of 50 words using a pre-trained bert-base-uncased tokenizer.[9] Then we extracted input_ids from the tokenizer that allows BERT to locate the position of words in sentences.
The tokenized corpus generates masks and segments, which help BERT to identify and understand contextual information. The input_ids, masks, and segments are used as inputs for the pre-trained BERT feature matrix.

4.2.6 SentenceBERT

With a similar procedure to BERT, we used the pre-trained transformer multi-qa-distilbert-cos-v1 to transform our corpus into a feature matrix.[10] The model was then passed through a fine-tuned sentence transformer to produce a feature matrix of sentence-specific embeddings.

4.2.7 BlueBERT

We were also interested in running BlueBERT, which was pre-trained and fine-tuned on 400 million words of PubMed abstracts and MIMIC-III clinical notes.[11] Unfortunately, the implementation was not possible due to the lack of computational resources.

4.3 Learning Models

We utilized multiple learning models for our exploratory research. In this section, we will introduce and describe our usage of the models. To prepare the data for binary classification, we split the feature matrices and target into random train and test subsets of 70%-30% split respectively. The features and targets from the training set were used to train models such as Logistic Regression, Naive Bayesian, Decision Tree, Random Forest, Support Vector Machines (SVM), and Long short-term memory (LSTM). Features from the test set were then fitted into the trained models to produce predictions.

4.3.1 Logistic Regression

Logistic regression is a simple method used for binary prediction that uses statistical analysis[12] Logistic regression model predicts a dependent data variable by analyzing the relationship between the independent features and predicts a variable that is dependent on the independent features.[12] In our approach, we simply used the converted feature matrix from our feature extraction methods and used it as our input for the model.

4.3.2 Naive Bayes

Naive Bayes classifier is a probabilistic machine learning model. Naive Bayes is primarily used for classification tasks. The classifier is based on the Bayes theorem which is expressed in equation (13)

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \quad (13)$$

Bayes theorem allows us to determine the probability of A happening given B has already happened. B is called evidence. Bayes classifier assumes that features are completely independent. That is why it is titled Naive Bayes. In our approach, we simply used the converted feature matrix found from our feature extraction method and used it as our input for the model.

4.3.3 Decision Tree

The decision tree classifier creates the classification model by building a decision tree. Each node in the tree specifies a test on an attribute, and each branch descending from that node corresponds to one of the possible values for that attribute. During training, the model learns the classes of training data by traversing from the root to the leaf of the tree. Leaves represent class labels of particular data. Similar to the other simpler models, we used the feature matrix converted from the corpus directly as the training and testing data of the model.

4.3.4 Random Forest

Random Forest is an ensemble method that consists of a large number of decision trees. The model is characterized as an ensemble classifier because it combines the results of several, therefore reducing the possibility of overfitting. The tree classifiers are generated randomly and also, provided randomized and dropout input in order to simulate the regularization effect. Similar to the other simpler models, we used the feature matrix converted from the corpus directly as the training and testing data of the model.

4.3.5 Support Vector Machine(SVM)

Support Vector Machines can be used for both classification and regression problems. SVM algorithm aims to create the decision boundary that can differentiate n-dimensional space into different classes so that new data points can be classified simply by putting them in the correct category in the future. This best decision boundary is called a hyperplane.

VM chooses the support vectors that assist in creating the hyperplane. That is why the algorithm is called Support Vector Machine. Fig. 4 displays the hyperplane functionality of SVM. Similar to the other simpler models, we used the feature matrix converted from the corpus directly as the training and testing data of the model.

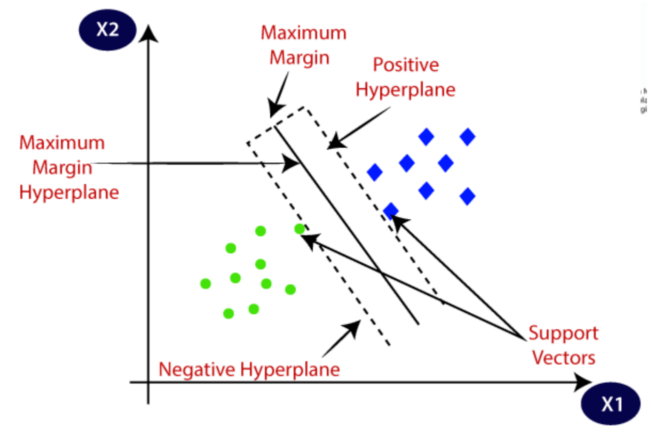


Fig 4. SVM Functionality

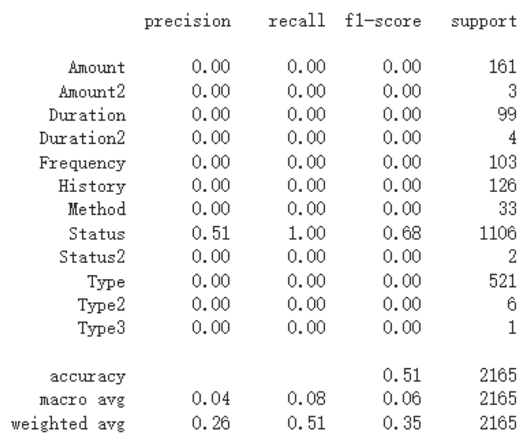
4.3.6 Long Short Term Memory Network (LSTM)

Long Short Term Memory Network(LSTM) is an advanced recurrent neural network(RNN). It is a sequential model that allows information passed through each LSTM cell to persist for some cycles. It is capable of handling the vanishing gradient problem faced by regular RNN. For our experiment, we used the weights of Word2Vec or the SentenceBert models to initialize the weights of the LSTM model and then used the feature matrix as input. We used global pooling and dense layers to get the outputs.

4.4 Binary vs Multiclass Classification

We decided to limit our scope to binary classification since our initial experiments with multiclass classification ran into severe issues of overfitting. Fig. 5 displays the performance of SentenceBERT embeddings on SVM with fine-tuned hyperparameters for multiclass classification.

The metric report demonstrated very good performance on "Status," while returning 0 on all other relation types. This means that the feature matrix is massively skewed towards one feature, whose performance is overblown and in turn returns a biased accuracy. To limit noise, we will only select the two most numerous two relation types (Amount and History) for the rest of our feature experiments, which allows us to concentrate on models that can truly differentiate labels instead of reproducing noise.



```
              precision    recall  f1-score   support

      Amount       0.00      0.00      0.00       161
     Amount2       0.00      0.00      0.00         3
    Duration       0.00      0.00      0.00        99
   Duration2       0.00      0.00      0.00         4
   Frequency       0.00      0.00      0.00       103
     History       0.00      0.00      0.00       126
      Method       0.00      0.00      0.00        33
      Status       0.51      1.00      0.68      1106
     Status2       0.00      0.00      0.00         2
        Type       0.00      0.00      0.00       521
       Type2       0.00      0.00      0.00         6
       Type3       0.00      0.00      0.00         1

    accuracy                           0.51      2165
   macro avg       0.04      0.08      0.06      2165
weighted avg       0.26      0.51      0.35      2165
```

Fig 5. The overfitting issue of multiclass classification

5. RESULTS

The resulting predictions from the test set of each model were then compared with the test set's targets to obtain accuracy scores for the models. The results can be seen in Table 1.

Model	Bag of Words	TF-IDF	Word2Vec	SpaCy + Word2Vec	BERT (Word)	Sentence -BERT
Logistic Regression	0.485	0.485	0.47	0.54	(b)	0.8
Naive Bayesian	0.493	0.49	0.51	0.6	(b)	(c)
Decision Tree	0.602	0.582	0.63	0.47	(b)	0.72
Random Forest	0.711	0.692	0.56	0.53	(b)	0.81
SVM	0.739	0.485	0.51	0.51	(b)	0.74
LSTM	(a)	(a)	0.49	0.5	(b)	0.5
Pre-trained BERT	(b)	(b)	(b)	(b)	0.61	(b)

Table 1: Results for each model

Looking at the accuracy scores for the binary models, the Sentence-BERT model has the best overall performance with accuracy scores ranging from 0.8 to 0.5. In the simpler models, Support Vector Machines using the Bag of Words approach obtained the highest accuracy at 0.711. The next best performing model was Random Forest using the Bag of Words approach with an accuracy of 0.711, followed by Random Forest using TF-IDF approach at 0.692.

The simpler models performed relatively well since our dataset is small. More complex and data-hungry models like SVM and LSTM tend to overfit, which accounts for their worse performance than simpler and more explainable models. For the real-world application, we expect the more complex models to perform better. Sentence-BERT is capable of getting more contextual information about the sentence. This fine-tuned BERT model computes the cosine similarity of the pair of sentence embeddings to represent accurately the semantic similarity of the two sentences.[14] This improvement is clearly reflected in our results.

6. FUTURE WORKS

Our experiments showed us that creating a model directly to predict the relationships present in the sentence is somewhat premature. We can make the model into a multimodal one where the model would first try to predict the entities of a model, then it will try to predict the relation between the entities.

We can also add the entities of the sentences as additional features to provide the model with a better context on the target.

We can also add the complete text as a different feature for the model. This may help the model gain a more contextual understanding. We will also incorporate contextual word embeddings like GloVe.

7. CONCLUSION

In this research, we focused on exploring different approaches to extract information on relation types from MIMIC-III medical notes. We used annotated medical notes provided in the SHAC dataset and preprocessed the dataset into the appropriate format. Then, we applied multiple feature extraction methods, designed multiple different models, and tested different combinations of feature extraction methods and models. We discovered that due to the size of the dataset, simpler models seem to have the best overall performance.We also found that Sentence-Bert greatly improves the performance of the models.

8. ACKNOWLEDGEMENT

This project would not have been possible without the support and mentorship from Dr. Yifan Peng and Song Wang. We are grateful to have the pleasure to work with them. Their supervision and recommendations have been invaluable in helping us work through the issues we encountered throughout the project.

We would also like to express our heartfelt gratitude to our professor, Dr. Ying Ding, for her guidance and insights during the course of this project. It was thanks to her introduction that we were able to collaborate with Dr. Yifan Peng on this project.

9. REFERENCES

[1] Kevin Lybarger, Mari Ostendorf, and Meliha Yetisgen. 2021. Annotating social determinants of health using active learning, and characterizing determinants using neural event extraction. Journal of Biomedical Informatics 113, (2021), 103631. DOI:https://doi.org/10.1016/j.jbi.2020.103631

[2] Pontus Stenetorp et al. 2012. BRAT: a Web-based Tool for NLP-Assisted Text Annotation. DOI:https://doi.org/10.5555/2380921.2380942

[3] Gehrmann, F. Dernoncourt, Y. Li, E.T. Carlson, J.T. Wu, J. Welt, J. Foote Jr,E.T. Moseley, D.W. Grant, P.D. Tyler, et al., Comparing deep learning and con cept extraction based methods for patient phenotyping from clinical narratives,Public Libr. Sci. One 13 (2) (2018) e0192360, http://dx.doi.org/10.1371/journal.pone.0192360

[4] Y. Wang, E.S. Chen, S. Pakhomov, E. Arsoniadis, E.W. Carter, E. Lindemann,I.N. Sarkar, G.B. Melton, Automated extraction of substance use informationfrom clinical texts, in: AMIA Annual Symposium Proc., 2015, pp. 2121–2130,https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4765598/

[5] M. Yetisgen, L. Vanderwende, Automatic identification of substance abuse fromsocial history in clinical text, Artif. Intell. Med. (2017) 171–181, http://dx.doi.org/10.1007/978-3-319-59758-4_18.

[6] Bag of words (BoW) model in NLP - GeeksforGeeks. GeeksforGeeks. Retrieved May 12, 2022 from https://www.geeksforgeeks.org/bag-of-words-bow-model-in-nlp/

[7] TF-IDF Explained And Python Sklearn Implementation | by Marius . Medium. Retrieved May 12, 2022 from https://towardsdatascience.com/tf-idf-explained-and-python-sklearn-implementation-b020c5e83275

[8] Embeddings, Transformers and Transfer Learning · spaCy Usage . spaCy. Retrieved May 12, 2022 from https://spacy.io/usage/embeddings-transformers/

[9] bert-base-uncased · Hugging Face. Hugging Face. Retrieved May 12, 2022 from https://huggingface.co/bert-base-uncased

[10] sentence-transformers/multi-qa-distilbert-cos-v1 · Hugging Face. Hugging Face. Retrieved May 12, 2022 from https://huggingface.co/sentence-transformers/multi-qa-distilbert-cos-v1

[11] bionlp/bluebert_pubmed_uncased_L-24_H-1024_A-16 · Hugging . Hugging Face. Retrieved May 12, 2022 from https://huggingface.co/bionlp/bluebert_pubmed_uncased_L-24_H-1024_A-16

[12] Rohith Gandhi. 2018. Naive Bayes Classifier – Towards Data Science. Medium. Retrieved May 12, 2022 from https://towardsdatascience.com/tagged/naive-bayes-classifie

[13] George Pounis. 2019. Statistical Analysis of Retrospective Health and Nutrition Data. Analysis in Nutrition Research (2019), 103-144. DOI:https://doi.org/10.1016/b978-0-12-814556-2.00005-1

[14] Richer Sentence Embeddings using Sentence-BERT — Part I | Genei. Medium. Retrieved May 12, 2022 from https://medium.com/genei-technology/richer-sentence-embeddings-using-sentence-bert-part-i-ce1d9e0b1343