Are Sequential Embeddings Useful for Predicting Neighbourhood Census Changes?

1. Abstract

In modern urban areas, many people tend to use Yelp to review local businesses and share their impressions and experiences. Studies have shown that the reviews on businesses is deeply correlated with its geographical area, the neighbourhood: different neighbourhoods have certain characteristics that allow them to be distinguished by business review embeddings in the area [1]. This project aims to compare the census change prediction performance between review vectors with and without sequential representations. As specific applications, the neighbourhood from the Greater Toronto Area between 2011 and 2016 are studied. The neighbourhood reviews are represented in three different ways: training an Autoencoder for embedding TF-IDF vectors without sequential properties, training an RNN encoder-decoder to for embedding the review dataset with sequential properties, and using the pre-trained ELMO for embedding reviews with sequential properties. To validate the first two self-trained models, the dataset is divided into training and testing sets, the training and validation loss curves are studied, and the testing loss is examined to see if the model generalizes. It was discovered that the TF-IDF Autoencoder has decent training results, while a vanilla RNN encoder-decoder structure built with LSTM could not embed Yelp reviews with sequential properties. Then, to evaluate the census change prediction performance of the Autoencoder and ELMO, multi-target linear and non-linear regressors are trained to predict Education and Income level proportion changes, with the review embeddings as inputs. For each of the evaluation regressors, the mean total absolute error for the prediction is calculated and compared. It was found that sequential embeddings are useful for predicting Education changes, while TF-IDF embeddings are sufficient for predicting Income changes.

2. Motivations

Previously during my thesis studies, I found that Autoencoders trained on TF-IDF review word vectors are predictive for neighbourhood census changes. This project aims to discover if sequential properties in the reviews could help improve the census change prediction result, because for many NLP applications, such as some sentiment classifications, sequential embeddings have better performance than TF-IDF encodings [2]. The pretrained ELMO used to be the state-of-the-art word representation in early 2018, outperforming a lot of traditional NLP models on different tasks. It is trained by Google using expensive computational resources on a huge text corpus, thus, it produces reliable word representations that have great performance across a broad range of NLP tasks [3]. A simple RNN encoder-decoder model has the potential to have better performance over the pretrained ELMO, because it will be trained on the Yelp review dataset, which is tailored for the specific application of this project.

The contributions of this project could help social-linguistic researchers develop a computational method of studying related topics, by leveraging existing data. Traditionally, these studies were done by conducting sociolinguistic interviews, which is very time and human resource consuming. With a computational method that is able to perform analysis based on existing datasets, the cost is greatly reduced, and the efficiency is largely increased [4].

3. Background

Word Embeddings

Word embedding is a representation of document vocabulary: they are vector representations of words, learned from different methodologies. Some common methodologies include Bag Of Words (BoW), representation or Term Frequency-Inverse Document Frequency representation of words, which essentially are vectors of words based on frequencies of one word occurring in a document. More advanced methodologies include neural word embeddings, which trains text inputs into vector outputs with neural networks. These embeddings are able to capture the similarity of words within a context [5]. For example, in Figure 3.1, the word "toilet" and "bathtub" would be very close to each other in the embedding vector space, because they are often used in a similar context [6]. The most commonly used models are Word2Vec, FastText and GloVe.

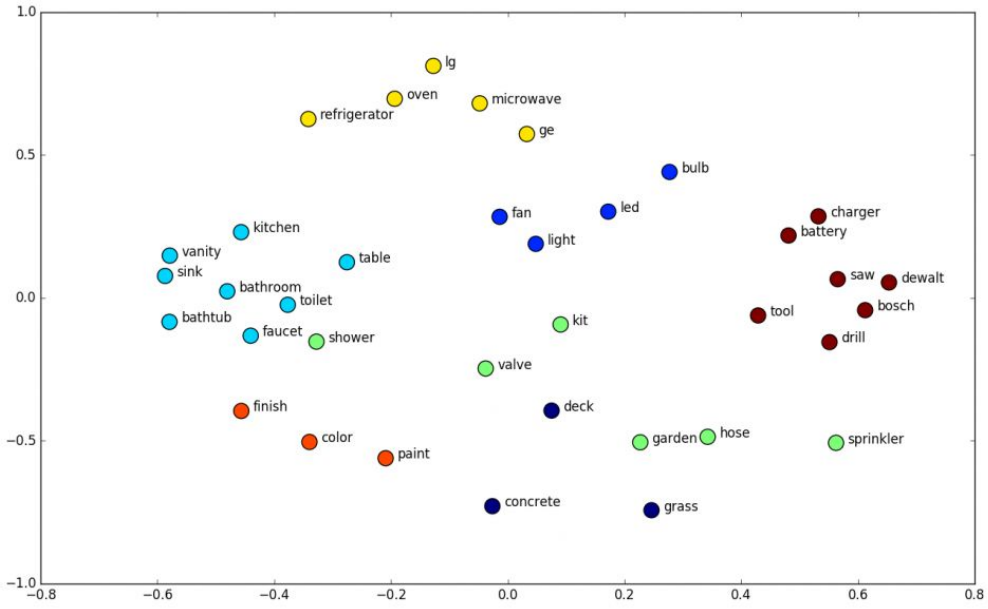

Figure 3.1. Illustration of Neural Word Embedding vector space with Word2Vec [6]

Recurrent Neural Network (RNN)

Recurrent Neural Network (RNN) is a modified version of basic neural networks, to work with serial inputs with no predetermined limit on size. RNN is able to remember things it learned previously, and pass that knowledge on while learning the next thing; that is, the output of the RNN is not only affected by its own corresponding weighting, like that of a regular NN, but is also affected by a hidden state vector representing context based on previous inputs and outputs [7]. As

illustrated in Figure 3.2, the output y is affected by weighting learnt from the input x (w_x), as well as weighting generated from a hidden state (h), w_h, that is passed from a pior unit. The two pieces of information together form the final weighting w_y. Because of its chain-like nature, it is widely adapted in Natural Language Processing (NLP) models.

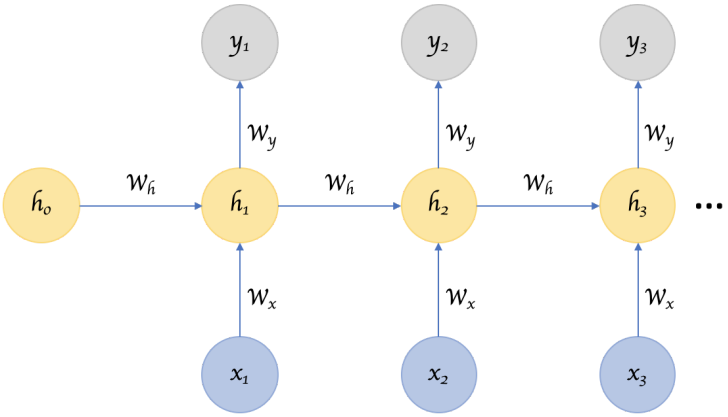

Figure 3.2. Standard RNN architecture [7]

Long Short Term Memory networks (LSTM)

Long Short Term Memory networks (LSTM) are a special kind of RNN that is capable of learning long-term dependencies. A standard RNN can remember prior information that is recent, however, it can not capture information from further back, which could be useful in many cases. LSTM is designed to conquer this issue. Compared to a standard RNN (Figure 3.3) that has a very simple structure, such as having a single tanh layer to generate a hidden state in each unit, a LSTM has a more complex structure with four layers interacting in a special way (Figure 3.4)..

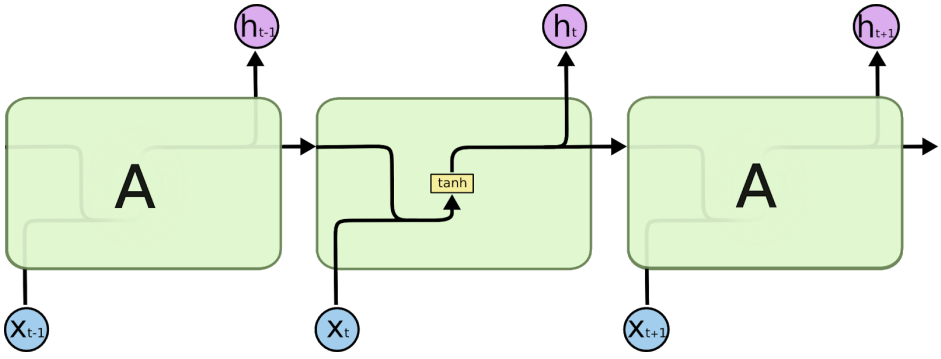

Figure 3.3. Standard RNN architecture [7]

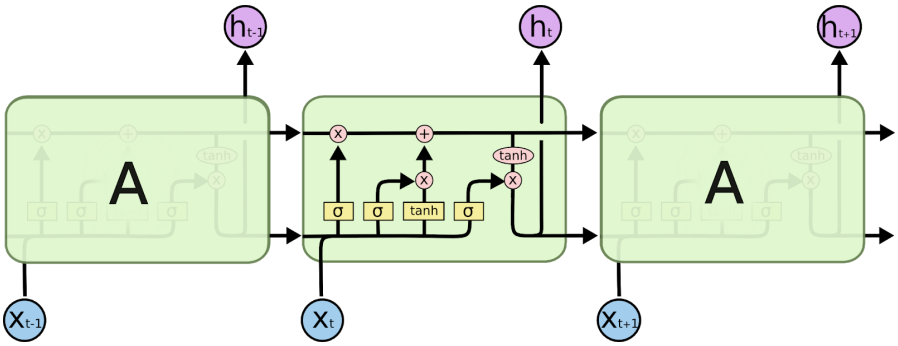

Figure 3.4. LSTM architecture [7]

Embedding from Language Models (ELMo)

ELMo representation of text corpus was published in early 2018. It is a type of deep contextualized word representation that is able to embed both complex characteristics of word use, such as syntax and semantics, as well as how these uses differ among linguistic contexts. The word embedding vectors produced by the model are derived from a Long short-term memory (LSTM) model that is pre-trained on a large text corpus. ELMo's representations are functions of all the internal layers of the bidirectional LSTM (biLM), instead of only the top layers, hence being 'deep' [3][8]. The architecture of the model is illustrated as in Figure 3.5, where the inputs, E, are embeddings of words from the same sentence. On the left, each embedding is passed to two LSTM layers, which are linked to each other in a sequence corresponding to the order of words in the sentence; on the right, each embedding is again passed to two LSTM layers, but these are linked in reverse order of the words in the sentence. Together, the neural nets on the left and on the right produce the final vector representation of each word in the sentence, T.

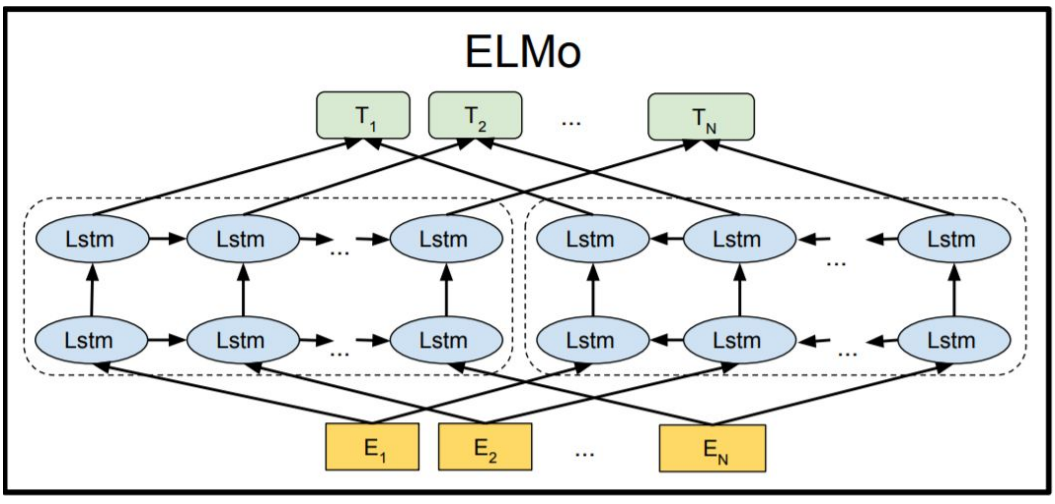

Figure 3.5.: ELMo architecture [8]

4. Data

4.1. Dataset descriptions

Yelp Business Reviews

The Yelp business reviews data is collected exclusively for the city of Toronto. The dataset includes approximately 24,000 businesses and 500,000 reviews [2]. The businesses in this dataset comprises different types of business, including restaurants, service business such as hair salons, entertainment services such as rock-climbing clubs, and many more. Each review sample in the dataset has a Business ID, a UserID, and a location of the business. The reviews span across 9 years, from 2009 to 2019.

Census Data

The Toronto Neighbourhood census datasets are collected by the City of Toronto [9]. The datasets were collected every 5 years, available to the public from 2006 to 2016. To match the Yelp review chronologically, the datasets in 2011 and 2016 are selected for analysis. There are many different topics in the neighbourhood census datasets, including 'Population', 'Land Area', 'Income', 'Education', 'Language spoken most often at home', and more. Some topics are divided into different categories, and the population under each category is recorded. For example, the language-related topics are divided into over 90 different languages, the income categories are divided into different income ranges, and education is divided into degree types. In total, there are 140 neighbourhoods in Toronto.

4.2. Data cleaning process

Yelp Business Reviews

Each model has a different data cleaning process, described under the following subsections,

TF-IDF Autoencoder

The data cleaning process for Yelp Business Reviews follows the methodology described in "Reading the city through its neighbourhoods" [2].

Tokenization

To handle misspelt words and incomplete sentences in the review texts, a natural language processing tool, spaCy, was used to extract standard word tokens from reviews. Then, accents were

removed from words, and stop words were removed from the token collection. Next, any words that occur in less than 100 distinct reviews were deleted, because they are very specific to certain businesses, and do not generalize. Next, some words are combined together to form 2-grams, such as “ice cream” and “pad thai”, which are more meaningful as a pair. These pairs of words are considered as one token, or feature. Finally, 1024 tokens with the highest frequencies of occurrence are selected - they account for approximately 70% of all words used in reviews. To create review vectors, the TF-IDF value for each token in each sample is calculated, and stored into a sample column. The final dataset is a 2D matrix, with each row representing a single review, and each column representing the TF-IDF value for a word.

Aggregation

For data aggregation, three different text corpuses were created, leveraging at business, user and neighbourhood levels respectively. For each of the corpuses, all TF-IDF review vectors are grouped by the attribute (business ID, user ID, or location corresponding to a neighbourhood), and the sum of TF-IDF values were taken. Now, there are 4 different text corpuses in total: R, the set of individual review vectors; B, the set of review vectors aggregated by business ID; U, the set of review vectors aggregated by userID, and N, the set aggregated by neighbourhoods. The R, B and U corpuses are concatenated to train Autoencoders for embedding reviews, and the N corpus is used for model testing. This particular dataset is not divided into train, validation and test set because all corpuses contain the same set of reviews aggregated in different ways. We only care about the reconstruction loss on the N corpus, which is used for neighbourhood census analysis, therefore, it is used for model testing.

Normalization

For each sample in each corpus, the vector is normalized such that the sum across the column is equal to 1. This cancels the effect of length of review text, which makes single review samples and aggregated samples comparable to each other.

RNN

Tokenization

For sequence representation trained with RNN, the reviews are first broken down into individual sentences. This is because the model was not able to learn reviews with hugely different lengths at all (i.e. the training loss does not decrease), and to improve efficiency in matrix storage (much less padding). The words in each of the sentences are turned into lower case, the punctuation indicating the end of the sentence is removed, because whether the sentence ends with a period ‘.’ or an exclamation mark ‘!’ does not affect the meaning of the sentence. The removal of the last punctuation improved the training result slightly. Then, each sentence was padded with a start of sentence token ‘<sos>’ and an end of sentence token ‘<eos>’. Finally, each sentence was padded with a padding token to match the length of the longest sentence. Out of all sentences, only tokens that have appeared more than 400 times are chosen to represent the sentences, out of all 3.6

million sentences in the training corpus. There are 4924 tokens selected in total. Each of the words in the sentence is represented with the pretrained GLoVE embedding of 50 dimensions. 50 was the smallest GLoVE representation, and it was chosen due to computational resource limits. Each unique token is assigned an index, later used for training.

Aggregation

To represent neighbourhood business reviews, all reviews in a neighbourhood would be broken down into individual sentences and embedded with the RNN model. The average embedding of all sentences in a neighbourhood will be the neighbourhood review representation.

ELMO

Tokenization

For ELMO, tokenization is built-in. Each review is passed into the model as a whole to obtain an embedding the size of sentence_length * 1024, which means that each word token in a sentence has a 1024 dimension.

Aggregation

To represent neighbourhood business reviews, all reviews in a neighbourhood are embedded with ELMO. Each review representation is taken as the average of all words in the review, and the neighbourhood representation is taken as the average of all reviews in the neighbourhood. The result is a neighbourhood embedding with 1*1024 dimensions.

Census data

For 2011 and 2016 data, under each topic, equivalent attributes are selected and matched, as shown in tables below,

Education

No certificate, diploma or degree
High school diploma or equivalent
Apprenticeship or trades certificate or diploma
College, CEGEP or other non-university certificate
University certificate or diploma below bachelor level
University certificate, diploma or degree at bachelor level or above

Income

Under $10,000
$10,000 to $19,999
$20,000 to $29,999
$30,000 to $39,999
$40,000 to $49,999
$50,000 to $59,999
$60,000 to $79,999
$80,000 to $99,999
$100,000 and over

As the first step for data cleaning, the topic and attribute names for Education and Income in 2011 and 2016 are matched. Compared to 2011, the census survey in 2016 is more comprehensive, and is divided into more categories. For example, the income ranges are divided into 17 categories in the 2016 survey, while there are only 9 in 2011. To account for the difference, the categories in 2016 are combined by taking the sum of populations, to match the categories in 2011. For each neighbourhood, the census data is normalized by dividing numbers for each category by their sum, making the sum of each row equal to 1. Therefore, each cell represents a proportion for the group of people in the total neighbourhood population. The final education dataset is a 140*6 vector, with 140 different neighbourhoods and 6 different categories, and the final income dataset is a 140*9 vector.

5. Methodology

General Approach

The general approach is broken down into two major parts, the first is to train embedding models on general Yelp reviews, and use it to obtain vector representations of neighbourhood-specific business reviews, which are referred to as neighbourhood embeddings in the rest of the paper. For the second part, the difference between neighborhood embeddings in different years are used directly as input into a linear regressor, which is trained to decode the neighborhood embeddings into a change in census proportions. The following sections describe detailed methodologies used for each part.

Generating Neighbourhood-Specific Business Review Embeddings

The methodology for generating neighbourhood-specific business review embeddings again is broken down into three parts, corresponding to different models.

Autoencoder

Method description

The autoencoder reduces the dimensionality of word vectors from 1024 to 16. The number 16 is chosen so that the dimension of features is significantly less than the number of samples (i.e. 140) to ensure model generalization. The model structure is illustrated in Figure 5.1, where a linear layer and an activation layer is added as pairs, and each pair has smaller dimensions than the previous pair in the Encoder. The decoder is a mirror of the Encoder, in terms of layer dimensions.

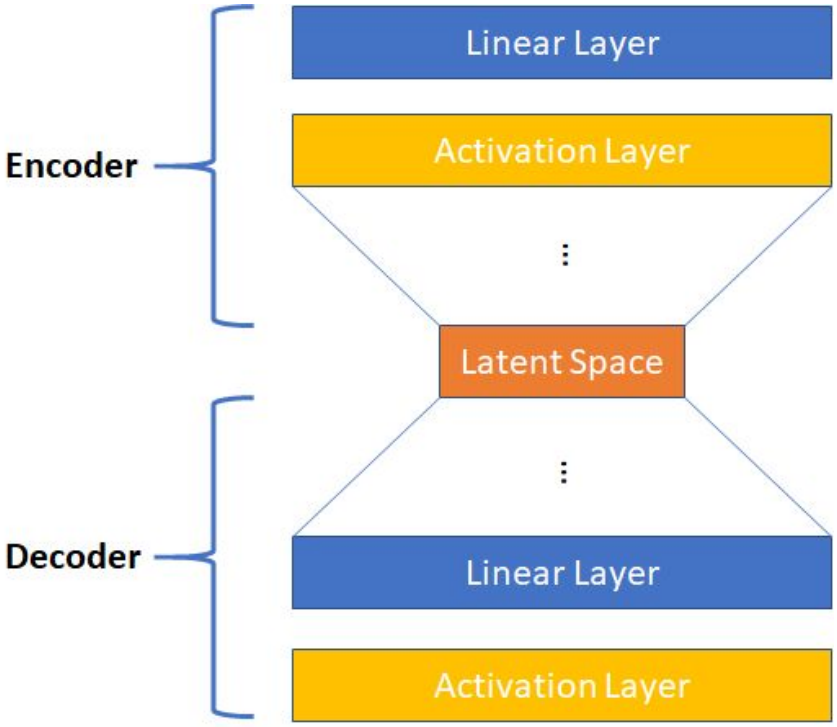

Figure 5.1. Autoencoder model structure illustration

To train the model, all four text corpuses described in the previous section are concatenated to create a large training set.

Model setup

Model architecture

Reviews Encoder

```
Encoder(
  (encoder): Sequential(
    (linear1): Linear(in_features=1024, out_features=1024, bias=True)
    (activation1): Tanh()
    (linear2): Linear(in_features=1024, out_features=256, bias=True)
    (activation2): Tanh()
    (linear3): Linear(in_features=256, out_features=16, bias=True)
    (activation3): Tanh()
  )
)
```

Reviews Decoder

```
Decoder(
  (decoder): Sequential(
    (linear1): Linear(in_features=16, out_features=256, bias=True)
    (activation1): Tanh()
    (linear2): Linear(in_features=256, out_features=1024, bias=True)
    (activation2): Tanh()
    (linear3): Linear(in_features=1024, out_features=1024, bias=True)
    (sigmoid): Sigmoid()
  )
)
```

Evaluation Criterion

Loss function for evaluating the model was selected to be Cosine-Similarity Loss. This particular loss function was chosen because it works particularly well for embedding tasks, as it handles sparse data well.

RNN

Method description

The RNN encoder-decoder model is adapted from an RNN sequence-to-sequence translation model [10][11]. The input is the sentence representation, containing indices of tokens at their corresponding positions, and it may contain padding tokens. When passed into the model, each token is represented with their GLoVE embeddings. In the original ‘Sequence to Sequence Learning with Neural Networks’ paper, the authors used a latent space dimension of 512. Because the task for this project is to encode and decode sequences in the same language, which is easier than the

task for the original paper, and with considerations of computational resources, our hidden dimension is chosen to be 256. The model structure is illustrated in Figure 5.2,

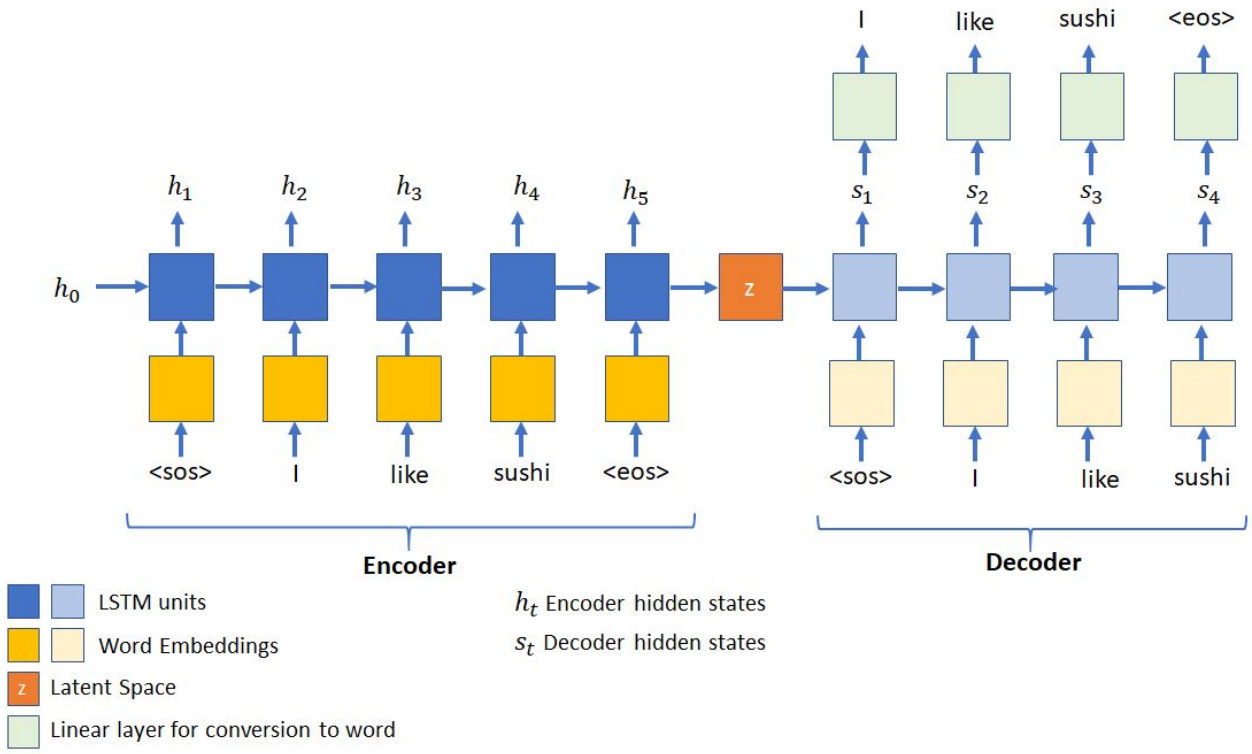

Figure 5.2. RNN encoder-decoder model structure illustration

As illustrated in Figure ?, each review sentence is passed through the word embedding layer, and inputted into the LSTM units in the Encoder. At each time step, the input into the LSTM is both the token embedding and the hidden state from the previous time step. The encoder outputs a sentence embedding, which is the latent space z. As for the Decoder, the input into the first LSTM is the latent representation z and generated embedding. Each LSTM then takes the hidden state from the previous LSTM and a new predicted word embedding. At the end, each hidden state is passed through a linear layer to be converted into actual word tokens [10].

Model setup

```
Seq2Seq(
  (encoder): Encoder(
    (embedding): Embedding(4924, 50)
    (rnn): LSTM(50, 256, num_layers=2, dropout=0.5)
    (dropout): Dropout(p=0.5, inplace=False)
  )
  (decoder): Decoder(
    (embedding): Embedding(4924, 50)
    (rnn): LSTM(50, 256, num_layers=2, dropout=0.5)
    (fc_out): Linear(in_features=256, out_features=4924, bias=True)
    (dropout): Dropout(p=0.5, inplace=False)
  )
)
```

Evaluation Criterion

Loss function for evaluating the model was selected to be Cross-Entropy Loss, because the input and output of the RNN model are indices of word tokens.

ELMO

As described in the Background section, ELMO is a pretrained model based on bidirectional LSTMs. The ELMO used for this project can be obtained from Tensorflow Hub [12]. The input to the model are strings of untokenized sentences, and the output is the weighted sum of the three LSTM layers of size [batch_size, max_length, 1024].

Decoding Neighbourhood Embeddings into Census Data

The end goal is to predict the change of proportions of census data in different categories for each of the 140 neighbourhoods, from the difference in embedding vectors between 2011 and 2016, as illustrated in Figure 5.3,

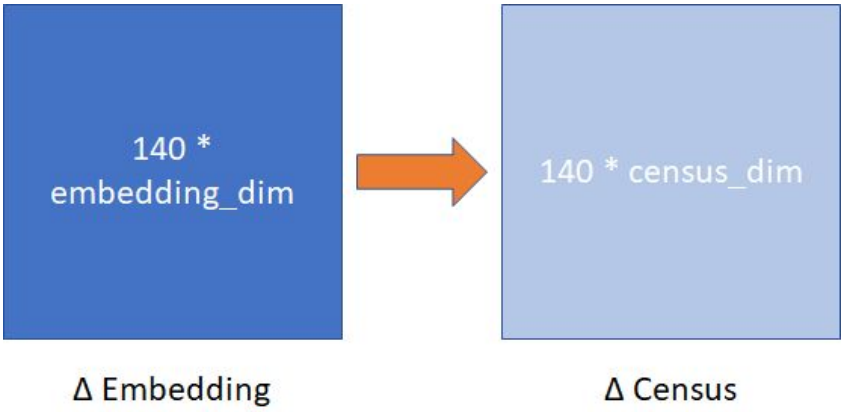

Figure 5.3. Census change prediction illustration

Each census topic has a decoder trained separately. To decode the neighbourhood embeddings into census data, for each validated model, both a linear and a non-linear regressor are trained. The training was done with nested-cross validation to ensure generalization across neighbourhoods.

Multi-Target Linear Regressor

Linear regression models are trained with Δ Embeddings as inputs and Δ Census as outputs. L2 regularization is used to prevent overfitting.

Multi-Target Non-Linear Regressor

The model is built in a way equivalent to the Decoder in an Autoencoder. It has linear layers and Tanh activation layers.

Evaluation Criterion

The testing results are evaluated by calculating the Mean Total Absolute Error for training and testing neighbourhoods:

$$MeanTotalAbsoluteError = \frac{1}{\#Neighbourhoods} \sum_{Neighbourhoods} \sum_{categories} |ActualProportion - PredictedProportion|$$

The census prediction performances are compared with the following two baseline methods:

- "No Change" Predictor, which simply returns the 2011 proportions as a prediction for the 2016 proportions
- Average Change Linear Regressor, which is a linear regression model, taking in delta neighbourhood review embeddings as input, predicting the average change across all categories.

6. Results and Analysis

6.1. Embedding Models

Autoencoder

The Autoencoder is successfully trained, with converging training and test losses, as shown in Figure 6.1,

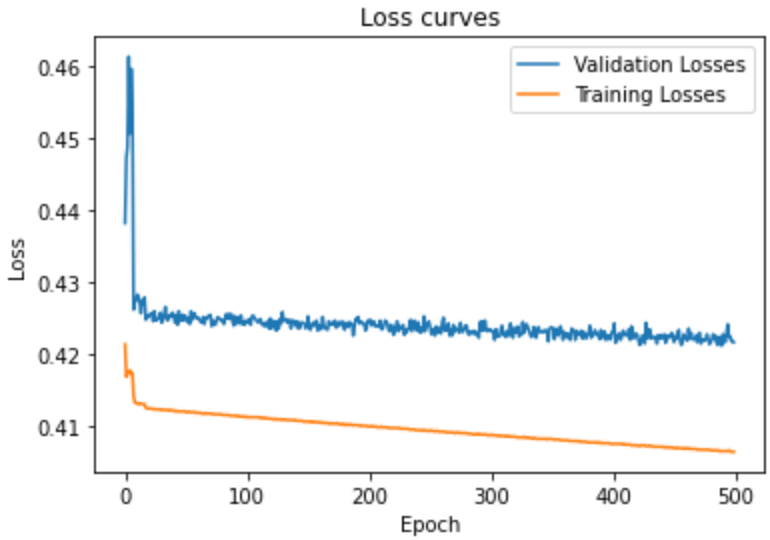

Figure 6.1. Loss Curves for Autoencoder

Although the model is overfitting, with validation losses higher than training losses, both losses decrease and converge to a plateau as epoch increases. This suggests that the model is able to learn from the dataset. Using Cosine Similarity Loss, a loss closer to 0 means their is 0 reconstruction error, while a loss closer to 1 means the reconstructed vector is completely different/orthogonal to the original input vector. Having a reconstruction loss of less than 0.5 means that the model is able to reconstruct the vectors with a considerable amount of accuracy, and the result is acceptable for the purpose of this project, as vector dimension is reduced from 1024 to 16.

RNN

Training on review sentences, the RNN encoder-decoder model is not able to learn successfully. As illustrated in Figure 6.2, even the training loss does not have a significant decrease in the first 5 epoches.

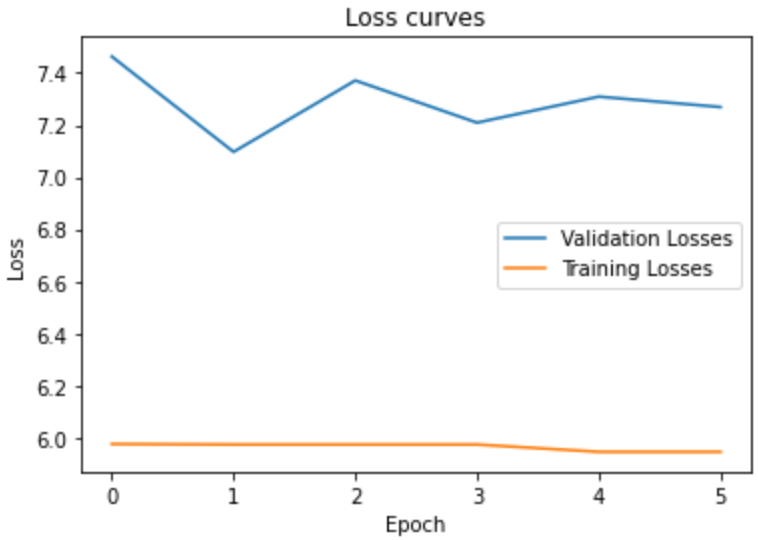

Figure 6.2. RNN encoder-decoder Loss Curves, trained on real data

However, the model training is successful when testing on a synthetic dataset. The synthetic dataset contains randomly generated review sentences from different words such as 'I really like food' and 'i don't like food'. As shown in Figure 6.3, both the training and validation losses decrease. Although the model is also overfitting after 10 epoches, both loss curves have a tendency to converge.

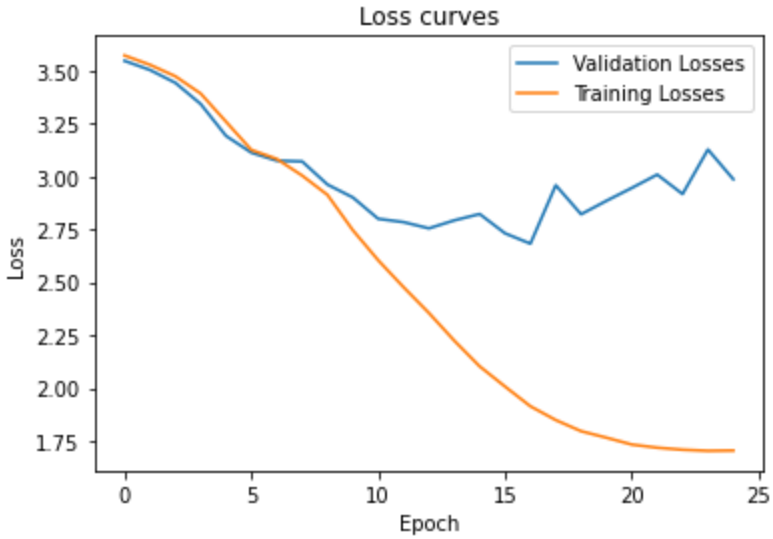

Figure 6.3. RNN encoder-decoder Loss Curves, trained on synthetic data

Because training on synthetic dataset is successful, we are sure that the model does not have a bug. Training fails on the real dataset could be caused by many reasons. One reason could be that the size of hidden dimensions (256) selected for training is not able to represent all parameters needed for the complex, real review embeddings.

Because the RNN encoder-decoder model is not used to predict census changes.

ELMO

The pretrained ELMO was successfully applied to the reviews to create embeddings, which are used to predict census changes.

6.2. Census Predictions

Embeddings generated with the TF-IDF Autoencoder model and ELMO model are used to predict census changes for Education and Income. Each embedding model is used to predict census with two different methods, Multi-Target Linear Regression (LR) and Multi-Target Non-Linear Regression with one layer (NLR1). Because the embedding dimension between ELMO embeddings and census is significantly different (1024 to 6 and 9), ELMO is also tested with Multi-Target Non-Linear Regression that has two layers (NLR2). The results are compared with a baseline "No Change" predictor, which is a dummy predictor that predicts zero change.

Education

The census prediction result for Education is summarized in Table 1,

Model	"No Change"	TF-IDF Autoencoder		ELMO		
		LR	NLR1	LR	NLR1	NLR2
Training MAE	11.92%	7.94%	8.22%	8.01%	5.62%	5.62%
Testing MAE	12.11%	10.07%	9.33%	8.08%	12.55%	9.25%

Table 1. Education Census Change prediction results with different models

From the results, we see that performing Linear Regression (LR) on ELMO embeddings yields the best prediction for census changes. Compared to the other models, it is not overfitting, and it has the best Testing MAE. The Non-Linear Regression with two layers (NLR2) has the second lowest testing MAE, however, it is overfitting, as the training MAE is significantly lower. These evidence suggest that sequential embeddings are useful for predicting Education census changes.

Income

Model	"No Change"	TF-IDF Autoencoder		ELMO		
		LR	NLR1	LR	NLR1	NLR2

Training MAE	14.76%	10.07%	9.89%	9.78%	7.94%	8.59%
Testing MAE	14.10%	10.32%	12.90%	10.63%	18.42%	13.10%

Table 2. Income Census Change prediction results with different models

For Income census change, the Linear Regression (LR) on TF-IDF Autoencoder has the best performance; it has the best testing MAE and is not overfitting. The LR on ELMO also has a decent result, however, the model is overfitting, and performs slightly worse than LR on Autoencoder. On the other hand, Non-Linear Regressions on ELMO perform very badly, both of them are overfitting, and are not performing well compared to the baseline model. This suggests that there is a linear relationship between review embeddings and Income changes, and that sequential embeddings are not useful for predicting Income changes.

Discussion

It is noticeable that embeddings with sequential properties are useful for predicting Education changes. On the other hand, TF-IDF embeddings without sequential properties are sufficient for predicting Income changes. This is understandable, as frequency of certain words could directly imply the income level in a neighbourhood, whereas implication for education level may require more complex information (encoded in sequential representations). Through sensitivity analysis for word features in TF-IDF autoencoder (using LR), we see that important features for indicating positive change in high income population include the increase of 'school' and 'class'. In comparison, important features for indicating positive change in low income population include the decrease of 'deep fried', and increase of 'sales', as illustrated in Figure 6.4.

```
chicken            0.000032
review removed     0.000020
highly recommend   0.000014
store             -0.000011
sashimi           -0.000011
salmon            -0.000010
school             0.000010
really good       -0.000009
burger            -0.000009
class              0.000008
Name:   $100,000 and over, dtype: float64
```

```
really good         0.000025
violating           0.000024
customer service   -0.000020
patio              -0.000014
amazing            -0.000014
sales               0.000014
customer           -0.000012
deep fried         -0.000012
fish               -0.000012
sashimi             0.000011
Name:   Under $10,000, dtype: float64
```

Figure 6.4. Important features for high-income population change (left), and low-income population change (right). Positive values indicate an increase in frequency leading to an increase in population, and vice versa.

This implies that, high-income neighbourhoods tend to have more services in the field of education, such as hobby classes and schools. The existence of these review words is enough to distinguish between neighbourhood income levels.

7. Conclusion and Next Steps

To conclude, this project aims to answer the question 'is sequential information useful for predicting census changes'. Through training and testing different models, including TF-IDF Autoencoder, RNN and pretrained ELMO, it was discovered that sequential information is useful for certain census topics such as Income, and not useful for others, such as Education. Embedding of Yelp reviews fails with RNN training, possibly due to insufficient hidden parameter sizes, while Autoencoder and ELMO were successful. As potential next steps, we could test on more census topics, to compare results and try to find a pattern for topics in which sequential information is useful.

8. References

[1] A. Besbes, “Overview and benchmark of traditional and deep learning models in text classification,” *KDnuggets*. [Online]. Available: https://www.kdnuggets.com/2018/07/overview-benchmark-deep-learning-models-text-classification.html. [Accessed: 05-Apr-2020].

[2] A. W. Olson, F. Calderon-Figueroa, O. Bidian, D. Silver, and S. Sanner, “Reading the city through its neighbourhoods: Deep text embeddings of Yelp reviews as a basis for determining similarity and change.”

[3] M. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, “Deep Contextualized Word Representations,” *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, Mar. 2018.

[4] D. Starks and Z. McRobbie-Utasi, “Collecting sociolinguistic data: Some typical and some not so typical approaches.” Simon Fraser University. [Online]. Available: https://www.sfu.ca/~mcrobbie/S.M.pdf

[5] Aaron and A. Bornstein, “Beyond Word Embeddings Part 2,” *Medium*, 03-Aug-2019. [Online]. Available: https://towardsdatascience.com/beyond-word-embeddings-part-2-word-vectors-nlp-modeling-from-bow-to-bert-4ebd4711d0ec. [Accessed: 05-Apr-2020].

[6] Shanelynn, “Get Busy with Word Embeddings – An Introduction,” *Shane Lynn*, 08-Feb-2018. [Online]. Available: https://www.shanelynn.ie/get-busy-with-word-embeddings-introduction/. [Accessed: 05-Apr-2020].

[7] M. Venkatachalam, “Recurrent Neural Networks,” *Medium*, 22-Jun-2019. [Online]. Available: https://towardsdatascience.com/recurrent-neural-networks-d4642c9bc7ce. [Accessed: 05-Apr-2020].

[8] Devlin, Jacob, Chang, Ming-Wei, Lee, Kenton, Toutanova, and Kristina, “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding,” *arXiv.org*, 24-May-2019. [Online]. DOI: arXiv:1810.04805

[9] Open Data Toronto, “Neighbourhood Profiles,” *City of Toronto Open Data Portal*, 07-Oct-2019. [Online]. Available: https://open.toronto.ca/dataset/neighbourhood-profiles/. [Accessed: 05-Apr-2020].

[10] B. Trevett, “Sequence to Sequence Learning with Neural Networks,” *GitHub*, 21-Jan-2020. [Online]. Available: https://github.com/bentrevett/pytorch-seq2seq/blob/master/1 - Sequence to Sequence Learning with Neural Networks.ipynb. [Accessed: 05-Apr-2020].

[11] I. Sutskever, O. Vinyals, and Q. V. Le, “Sequence to Sequence Learning with Neural Networks,” *Sequence to Sequence Learning with Neural Networks*, Dec. 2014. [Online]. DOI: arXiv:1409.3215

[12] Google, “TensorFlow Hub: elmo,” *TensorFlow Hub*, 04-Apr-2020. [Online]. Available: https://tfhub.dev/google/elmo/2. [Accessed: 05-Apr-2020].