UNSW SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

Word Embedding and Question Answering

Cecilia Xifei Ni z5173159

December 11, 2019

INTRODUCTION

"Who is the first Prime Minister of Australia?"

Figure 1 is a screen shot of search result returning by Google. This is a question a Question Answering (QA) system should be able to respond to. QA system evaluates texts across the web or database to find answer of a particular question to return in a form of short text.

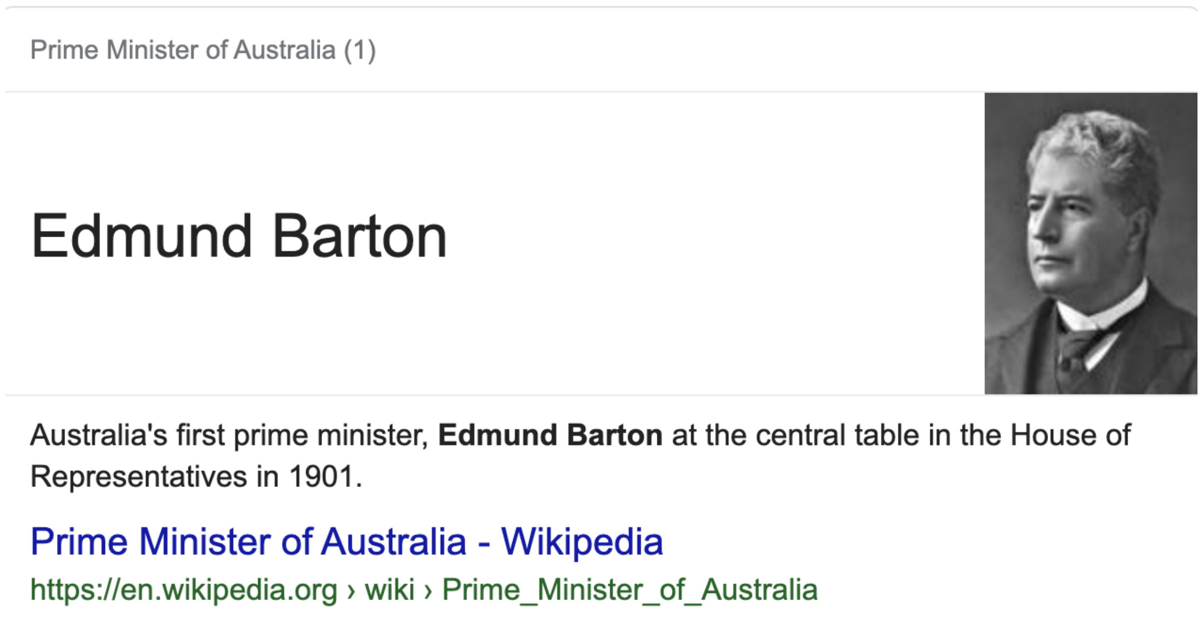
Current machine algorithms extracts answer from a short paragraph instead of a long content (e.g. an entire wiki page/a news article). When applying modern machine learning algorithms on a long content, the result can be complicated and lenthy.

This essay aims to firstly gives a mathematical heavy explanation on nowadays natural language processing (NLP) algorithms, then propose my own heuristics based on them to predict answers from a long content based on the question being asked.

My algorithm will be tested with Google Natural Questions which contains its own private testing dataset.

GENERAL WORD EMBEDDING MODELS

Word Embedding is the collective name for techniques in NLP where words or phrases are mapped into vectors of real numbers.



Prime Minister of Australia (1)

Edmund Barton

Australia's first prime minister, **Edmund Barton** at the central table in the House of Representatives in 1901.

Prime Minister of Australia - Wikipedia
https://en.wikipedia.org › wiki › Prime_Minister_of_Australia

Figure 1: A result returned by Google to answer "Who is the first Prime Minister of Australia?"

As per all machine learning algorithms, the general philosophy is to find the minimum of a converged function. When trying to convert words to number, its original utf code will not work well. What we need is a hard-coded map between the word itself, and its semantics. As human, we learn these semantics from daily experience which machine has no way to access, therefore, we need a mapping from words and its semantic in order for machine to decode.

First Attempt: Hot One Encoding

Given a collection of N unique words, each word is of size N. The vector is very sparse such that only one index is 1 and others are 0. This is to say each word is a **dimension**.

Figure 2 shows an example of hot encoding.

However, this model has some significant drawbacks. Firstly, it probably works fine for English as it has roughly 8000 frequent words. However, for languages like Chinese or Japanese, they are character-based, and each character will generate tons of combinations. This model will suffer from dramatic increase in dimensionalities. Secondly, all vectors in the matrix are independent to each other, while we do want them to learn the inter-connections between them.

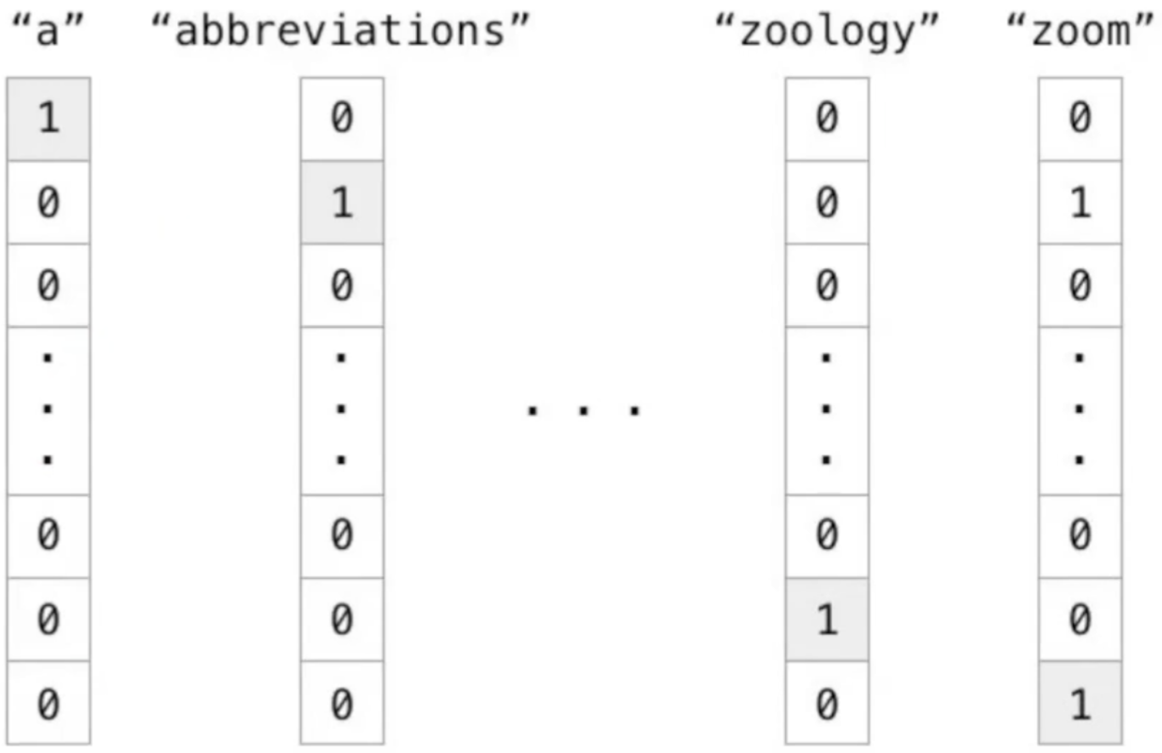


Figure 2: One Hot Encoding Example.

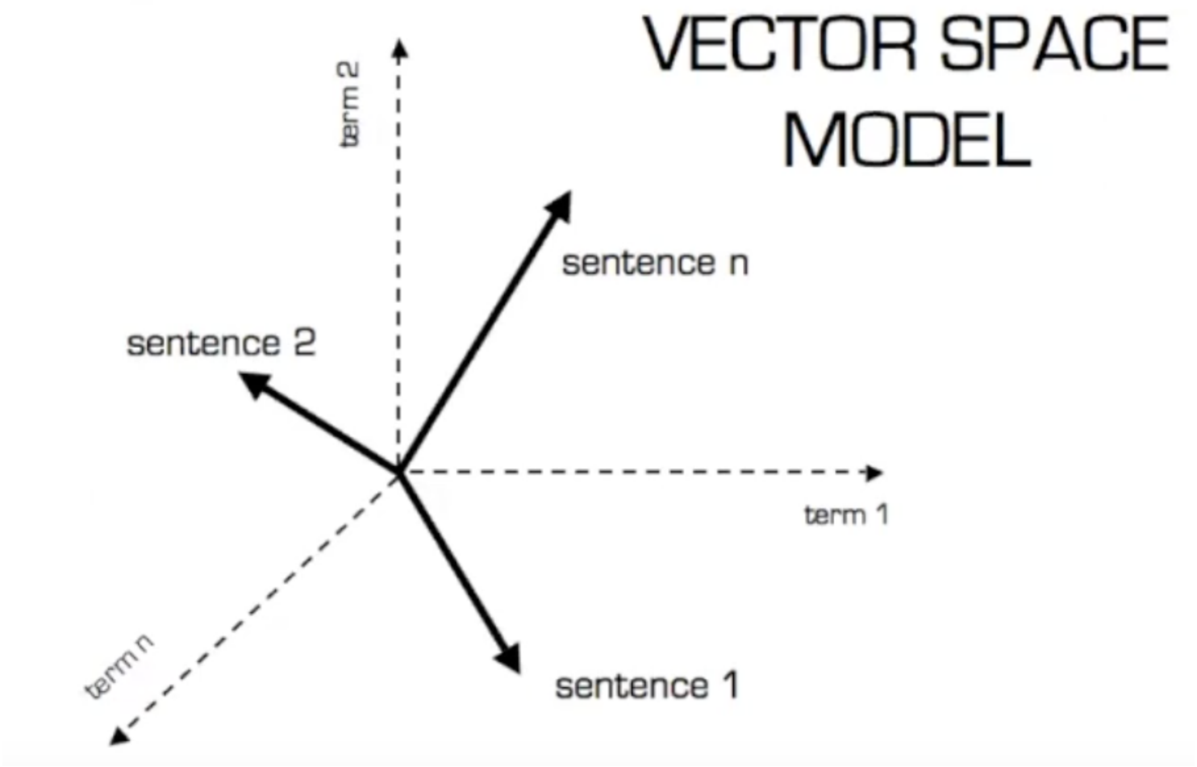


Figure 3: Bag of Words Vector Space Representation

SECOND ATTEMPT: BAG OF WORDS (BOW) MODEL

On the second attempt, we focus more on extracting information from the perspective of the whole document. Under this model, we represent words in a document as a bag (multiset) of words– we discard order and grammar, and only keep its multiplicity.

Here is an example:
Document 1: "George" "likes" "to" "play" "video" "games", "Mary" "likes" "video" "games" "too".
Document 2: "Mary" "also" "likes" "movies".

The BoW representation of the two above documentations is

BoW1: {"George" : 1, "likes : 2", "to" : 1, "play " : 2, "video" : 2, "games" : 2, "Mary" : 1, "too": 1}
BoW2: {"Mary" : 1, "also" : 1, "likes": 1, "movie" : 1}
Here, each key is the word, and the value corresponds to occurrence.
This approach could be potentially used to compare similarities between two documents. When comparing them, we construct two vectors with each index corresponding to each unique words. Each be the count of the word appeared in that specific document. The

	0	1
gender	male	female
age	child	adult

Table 1: Table 1: Represent Words in Fewer Dimensions

similarities of two documents can be measured by its Euclidean distance or its consine similarities.

Intuitively, two documents are similar if their vector points to the similar direction. We define the cosine similarities of two documents as:

$$cosim(U_k, U_i) = \frac{\langle U_k, U_i \rangle}{\|U_k\| \cdot \|U_i\|} \tag{1}$$

where $\langle U_k, U_i \rangle$ is the scalar product of U_i and U_k.

Third Attempt: Neural Network Based (NNLM)

One intuitive question to ask is, Bag of Words and One Hot Encoding both treat each words as a separate dimension, but do we need so many dimensions?

The answer is probably no. Because a lot of words are similar or related. e.g.

Nouns: dog, cat, pet.
Verbs: fish, fished, fishing.
Adjective: very, great, significant.
Context: play guitar, piano, games, tennis

To not feel overwhelmed by the math at first, let's start with a easy example:
Words to be processed: boy, girl, woman, man
We can simply divide these words into two dimensions: gender and age as per Table 1 shows. This method is called **Distributed Representation**.

However, human language are so complex that we are not able to pre-define all dimensions for all words. Therefore we want to train a function **f** to get the word embedding for us. Figure 4 shows the function mapping.

To the best of my knowledge, we can do this in two ways.

Autoencoder (GAN)

An autoencoder is a type of neural network that learns to label dimensions in an unsupervised manner. Figure 5 shows that an autoencoder is composed of two parts: the input

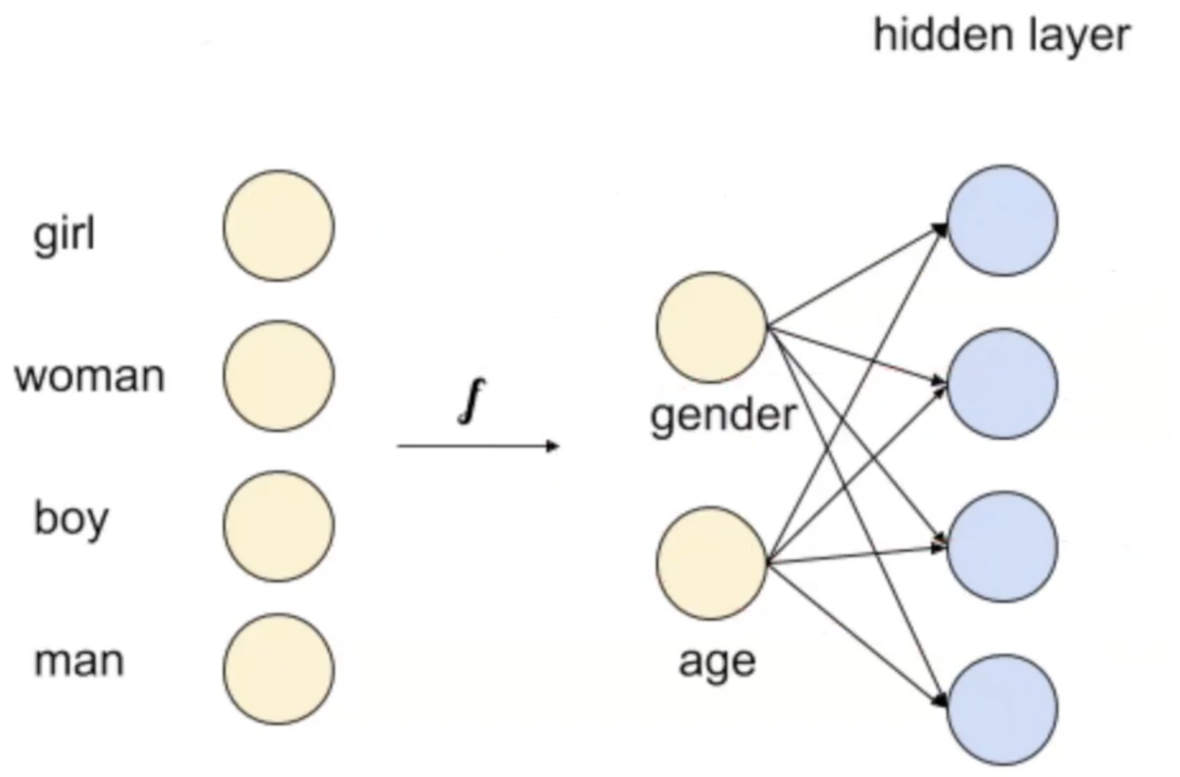


Figure 4: Figure 4: Neural Network Based Mapping

side, marked as X, and the reconstruction (output) side, marked as X'. The X tries to learn a distributed representation of the original data that aims to reduce the dimensions, and the X' side is where the autoencoder tries to generate from the reduced representation as close as possible to its original input, hence its name, reconstruction.

BASIC ARCHITECTURE OF AUTOENCODER An autoencoder consists of two parts, the encoder and the decoder, which can be defined as transitions ψ and ϕ, such that:

$$\begin{aligned} &\psi : \chi \mapsto \mathfrak{F} \\ &\phi : \mathfrak{F} \mapsto \chi \\ &\psi, \phi = arg_{\psi,\phi} min |\chi - (\psi * \phi)\chi| \end{aligned}$$

In the simplest form, we assume there is only one hidden layer. The encoder stage of the autoencoder takes input $x \in \mathbb{R}^d = \chi$ and maps it to $\mathbf{h} \in \mathbb{R}^p = \mathfrak{F}$.

$$\mathbf{h} = \delta(\mathbf{W}x + \mathbf{b})$$

This image $\mathbf{h}$ is the code section shown in Figure 5. δ is normally a sigmoid function defined as following:

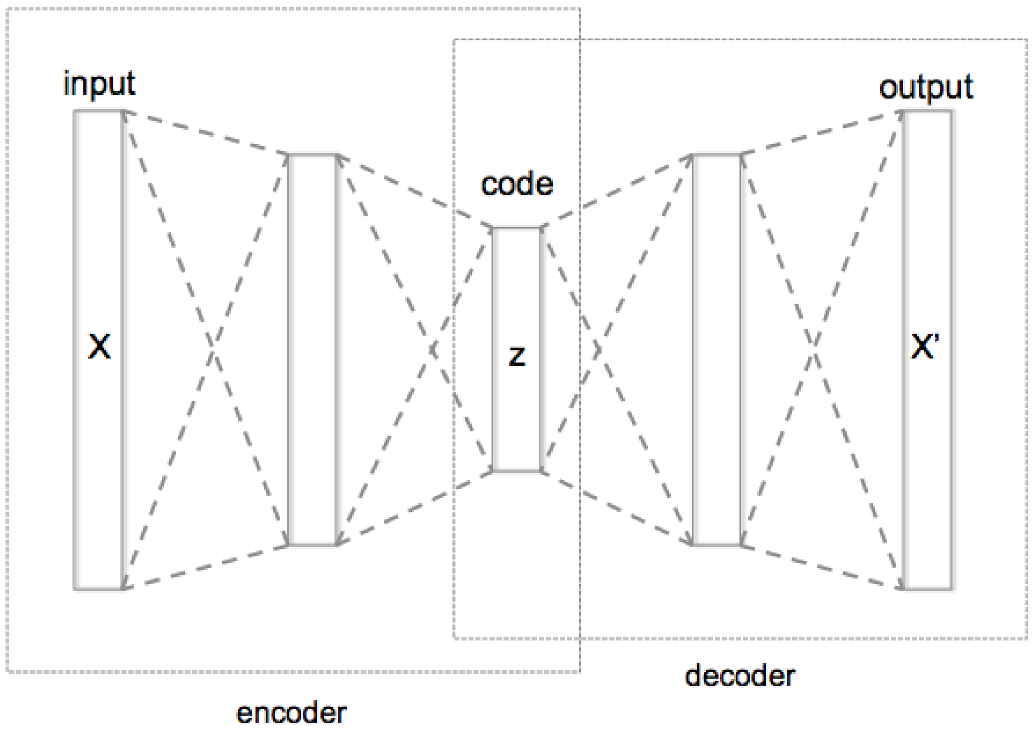


Figure 5: Figure 5: Schematic structure of an autoencoder with 3 fully connected hidden layers. The code (z, or h for reference in the text) is the most internal layer.

$$S(x) = \frac{1}{1+e^{-x}} = \frac{e^x}{e^x+1}$$

W is a weight matrix and **b** is a bias vector. The initializations of these two variables are random. They will be learnt iteratively through **backpropagation**.

$$\mathbf{h}' = \sigma(\mathbf{W}'x + \mathbf{b}')$$

The backpropagation computes weight space **W** and bias vector b with respect of a loss function $\mathbb{C}$.

x : input (vector of features).
x' : target output, the closer to x, the better.
$\mathbb{C}$: loss function, which intuitively associate with the "cost" of certain event or representation. The aim of training is to minimize the loss function.
L: the number of layers. In the above example, L equals to 1. However, here we aims to provide a generalized form.
$W^l = W^l_{j_k}$: the weights between layer $l-1$ and l, where $W^l_{j_k}$ is the weight between the k-th node in layer l-1 and the j-th node in layer l.
f^l: sigmoid function at layer l.
The whole neural network can be represented by this function:

$$g(x) = f^L(W^L f^{L-1}(W^{L-1} ... f^1(W^1 * x)...)$$

For each input x_i, there will be an output x'_i corresponding with it. the loss of the model on that pair is the cost of the difference between the predicted output $g(x_i)$ and the target output x'_i:

$$\mathbb{C}(y_i, f^L(W^L f^{L-1}(W^{L-1} ... f^1(W^1 * x)...))$$

For each layer, we compute the minimum of the loss function by calculating its derivatives.

$$\frac{\partial \mathbb{C}}{\partial W^l_{j_k}}$$

Where the loss function is defined as:

$$\mathbb{C}(x, x') = |x - x'|^2 = |x - \sigma'(\mathbf{W}'(\sigma(\mathbf{W}x + b)) + b'|$$

where x is the average across all inputs.

Neural Network Language Model (NNLM)

Fourth Attempt: Non-NN Model

References

[1] Prof. Mike Gal, "Physical waves", Lectures, *UNSW: PHYS1241*

[2] R Nave, ”Work Functions for Photoelectric Effect” *Hyperphysics* (2017), found at <http://hyperphysics.phy-astr.gsu.edu/hbase/Tables/photoelec.html>, accessed 26 Oct. 2017