# Comparing Different Models' Contextualization

Ziyou Hu

May 2023

## 1 Introduction

Transformer is a deep learning model that processes sequential data ("Transformer (machine learning model)", 2023). Transformer is often used is natural language processing, such as translation. Transformer is known for its ability to consider different parts of the input, known as self-attention mechanism. My group consists of Anh, Adi, Aidan, and me. In this project, we explored how different models contextualize meanings when they convert an input word in a sentence to a vector. In particular, we compared 2 models: CANINE and BERT. We feed input texts into the model and obtain the word vectors (with some adjustments based on the model). We designed a series of tests to evaluate how the models contextualize in terms of processing the order of the inputs, interpreting the punctuation, capturing information from other words, and so on.

## 2 Overview of Transformers

Transformer has an encoder and a decoder (Pietraho, 2023). An input sentence is first tokenized, which means breaking up the texts into small units, such as words or characters depending on the tokenizer ("Summary of the tokenizers", n.d.). Each unit is called a token. The tokenizer outputs each token as a vector (Pietraho, 2023). The length of the vector is the number of tokens of the tokenizer. All entries are 0 except for one entry, which has value 1 (Grg, 2019). Each token has a unique position for where the 1 is. This is called one hot encoding. Then, these vectors enter the encoders (Pietraho, 2023). The encoders encode each word as a vector and add in meanings from other words in the input. Then, the decoders use the outputs of the encoders to suggest texts. We would focus on the outputs of the encoders because we would like to assess how well the encoders embed contextual meanings into these vectors.

## 3 Tokenizers

A tokenizer breaks an input test into tokens ("Summary of the Tokenizers", n.d.). There are three main types of tokenization: Word-based tokenization, Character-based tokenization, and Subword tokenization.

**Word-based tokenization** breaks a text into words. Even if we consider the punctuation as an independent unit to eliminate words that will be created through combination with different punctuation, we will still have an enormous vocabulary size. Because the length of the vector is the number of tokens (Pietraho, 2023), each token vector will be very long. Thus, the embedding matrix will be enormous, which will use up large amount of memory and cause long run time (“Summary of the Tokenizers”, n.d.).

**Character-based tokenization** breaks a text into characters. It has a very small vocabulary size, but it is difficult to represent the meaning of single characters.

**Subword-based tokenization** keeps common words as a unit and views less common words as combinations of common words. Thus, the size of subword tokenizers are smaller than the size of word-based tokenizers. Also, subword tokenizers contextualize meanings better than character-based tokenization. Moreover, subword tokenizers can learn new words by breaking the new word into subwords that its knows.

## 4 The Mathematical Exploration of the Encoders

Both the encoders and decoders are a stack of layers (Alammar, n.d.). The number of encoders and decoders are the same. Each encoder has a self-attention layer, which embeds contextual meanings into each word, and then sends the outputs to a feed forward neural network. The first encoder in the stack receives the the token embeddings outputted by the tokenizer. For the rest of the encoders, they receive the outputs of the previous encoder.

### 4.1 Self-Attention Layer

First, we have our input matrix X. The vector of each word is a row in the matrix. The self-attention layer has multiple attention heads.

Suppose there are h attention heads. The following will happen for each

attention head i.
At each attention head i, there are three weight matrices: $W_i^Q, W_i^K, W_i^V$.
Multiply X by each of the 3 matrices to obtain 3 matrices: Query, Key, and Value.
Query: $Q_i = X \times W_i^Q$
Key: $K_i = X \times W_i^K$
Value: $V_i = X \times W_i^V$
Next, we obtain a $Z_i$ matrix by $Z_i = softmax(\frac{Q_i \times K_i^T}{\sqrt{d_k}})V_i$

### 4.1.1 Understanding $Z_i$

Suppose $\vec{q}$ is a row in the Query matrix $\vec{k}$ is a row in the Key matrix. Call the dot product $\vec{q} \cdot \vec{k}$ as a score that measures how much a word in the sentence matters to a word we are encoding. For example, $\vec{q_2} \cdot \vec{k_5}$ is the score of word 5 against word 2.
We calculate the scores for all words by taking the dot product of every row in Query and every row in Key. Because Q and K are matrices, we multiply Q by $K^T$. $Q_i \times K_i^T$ is a matrix where each row is the collections of scores for a particular word. For example, entries in row 5 measure how every word in the sentence matters to the fifth word.
Then, we divide $Q_i \times K_i^T$ by $\sqrt{d_k}$ to stabilize the gradient. $d_k$ is the dimension of each key vector, which is the length of every row in the Key matrix.
Next, we take softmax of $\frac{Q_i \times K_i^T}{\sqrt{d_k}}$ so that entries in each row is between 0 and 1, and add up to 1.
Therefore, $softmax(\frac{Q_i \times K_i^T}{d_k}) \times V_i$ adjust the values of $V_i$ by the level of importance of each word by context.

Finally, we concatenate all the $Z_i$ matrices to obtain a matrix $[Z_1Z_2...Z_n]$ and multiply the concatenated matrix by a weight matrix W to obtain a matrix $Z = [Z_1Z_2...Z_n] \times W$. We send Z into the Feed Forward Neural Network.

## 5 Experiment

The aim of the experiment is to compare how different models with different tokenizers contextualize meanings in word embeddings. The two models we used are CANINE and BERT. The experiment will test the two models' ability in their word embeddings from a variety of aspects. For example, adjust meanings by the order of the words, addition of information, interpreting punctuation, and so on. Because CANINE only produces character embeddings, we "manufactured" a word embedding in 2 ways, which will be described below. Because it is more difficult to add contextualize meanings into characters("Summary of the Tokenizers", n.d.), I hypothesized that the CANINE model is less capable of incorporating contextual meanings in its word embeddings.

### 5.1 CANINE Model

Jonathan H. Clark, Dan Garrette, Iulia Turc, and John Wieting proposed the CANINE model (Clark et al., 2022). The authors recognize that tokenization in common models limit the models' flexibility and makes the model less ideal for some languages. Instead, they train CANINE on character-level without explicit tokenization steps ("Canine", n.d.).

CANINE uses downsampling to reduce the length of input sequence. Downsampling is removing data points of the more abundant sample when there

is an imbalance of the class labels (Kumar, 2021). CANINE uses at least 3 encoders (“Canine”, n.d.). The inputs first enter the shallow encoder. A shallow encoder only has one layer. It uses attention to contextualize character embeddings. Then, CANINE applies downsampling. Next, the inputs enter the deep encoder and undergoes unsampling. The deep encoder is a equivalent to an encoder in BERT. Unsampling is the opposite of downsampling. It adds fake or copies of the less abundant sample when to offset the imbalance (Kumar, 2021). Finally, the shallow encoder creates the final character embeddings (“Canine”, n.d.).
Although CANINE is trained without tokenization steps, the use of tokenizer is optional. We used the tokenizer in our experiment. The CANINE tokenizer tokenizes the input into characters.

## 5.2 BERT Model

BERT stands for Bidirectional Encoder Representations from Transformers. This model is released by Google in 2018 (McCormick, 2019). It was trained on Wikipedia and Book Corpus.
BERT’s tokenization algorithm is WordPiece (“Summary of Tokenizers”, n.d.). WordPiece uses subword tokenization. During training, the vocabulary begins with all the characters in data. Then, WordPiece learns to combine characters based on likelihood. For example, WordPiece only merges symbol “x” and “y” if the probability of “xy” divides by the probability of “x” followed by “y” is greater than any other possible merges.
In the case of a new word, BERT tokenizer will decompose it to the largest component possible (McCormick, 2019).

## 5.3 Experiment Set-Up

My group sets up a Colaboratory notebook for each model. Both notebooks are adapted from Chris McCormick's notebook "BERT Word Embeddings v2.ipynb", which is available on his article "BERT Word Embeddings Tutorial" (McCormick, 2019). Both notebooks are set up to take 2 input sentences and run the inputs separately through the model.

We will extract word embeddings from BERT and CANINE and compare their cosine similarity. To avoid having to combine 2 subwords, most of the words we test are common, short words.

### 5.3.1 Notebook Setup

For **BERT**, we first install the Transformer and import the BERT model and BERT tokenizer (McCormick, 2019). We typed the inputs and add special tokens to the inputs. The special tokens are used to recognize the start ,"[CLS]", and the separation, "[SEP]", of sentences. Then, we used the BERT tokenizer to tokenize the texts.

Next, we ran the input through the BERT model and obtain the outputs. We obtained the hidden states of all the layers, which are the third item of the outputs. The hidden states are arranged as the number of 1) Layer, 2) Batch, 3) Token, and 4) feature. To extract word vectors, we needed to group the values by tokens. So, we rearrange the hidden states into 1) Token, 2) Layer, and 3) feature.

Now, each token has 13 vectors of $R^7 68$. To compare the word vectors, we summed up the last 4 layers to create a single vector, which is shown to be a good contextualized embedding. Other good contextualized embeddings are the second to last layer, concatenating the last 4 layers and so on.

Finally, we used cosine similarity to compare the two word vectors.

For **CANINE**, just as BERT, we first install the Transformer and import the CANINE model and CANINE tokenizer. We typed the inputs. Then, we convert the input texts into their labels (Lokare, 2023). In the code below, inputs1 is the input we would like to convert to their labels. encoding1 is the label it maps to. The code is adapted from the pose "CANINE" by Huggingface.

```
encoding1 = tokenizer(inputs1, padding="longest", truncation=True,
return_tensors="pt")
```

We ran the numeric representation (the tokenized input) into the CANINE model. Next, We grab the third item in the outputs, which is the hidden state (McCormick, 2019). We accessed the first batch of the second to last layer of the hidden state. Finally, we extracted the vector for each token and compare their cosine similarity. For example, if our sentence is "I read a book yesterday," and we want to get the vector for the character "r" in "read", we would get

```
x1 = hidden_states[-2][0]
vector_for_r = x1[2]
```

If we want to get the vector for "read", we add up the vectors for each character in the word "read".

We tried 2 approaches for comparing words in CANINE. The first approach is that we obtain a vector representation for the entire word by adding up the vector for character in each word. The second approach is that we just compare the vector of first character in each word. The advantage of the first one is we are able to obtain information from every character in the word.

The issue is we are not sure if the summation of the characters represents the meaning of a word. The advantage with the second approach is that we know the vector represents the character. The disadvantage is that we will lose meanings from other characters. We will compare these 2 approaches through this experiment.

### 5.3.2 Tests in the Experiment

The word being compared is in **bold**.

- **Past tense vs. Present tense**: Some verbs have the same spelling for past tense and present tense. The tenses can only be inferred by context. This test evaluates to what extent can models interpret nuanced temporal implication from other words and embed the temporal implication into a word embedding. The better performing model should have high but less than 1 cosine similarity, because the basic meanings of the words are still the same, except the time.
  - For example, “I **read** a book today.” versus “I **read** a book yesterday.”
- **Punctuation**: 2 sentences use exact same words but with 1 different punctuation that shifts meaning of the text. Punctuation is different from words and characters in that itself does not carry meanings. This is a much more difficult contextualization test. It evaluates whether the models can not only retrieve meanings from other words but also from punctuation. Better performing model should have lower cosine similarity.
  - For example, ”Let us hide, **Johnny**.” vs ”Let us hide **Johnny**.”

Johnny is the recipient in the first sentence, but Johnny is the person being hidden in the second example. The inspirations for the punctuation examples are from DigitalSynopsis.com.

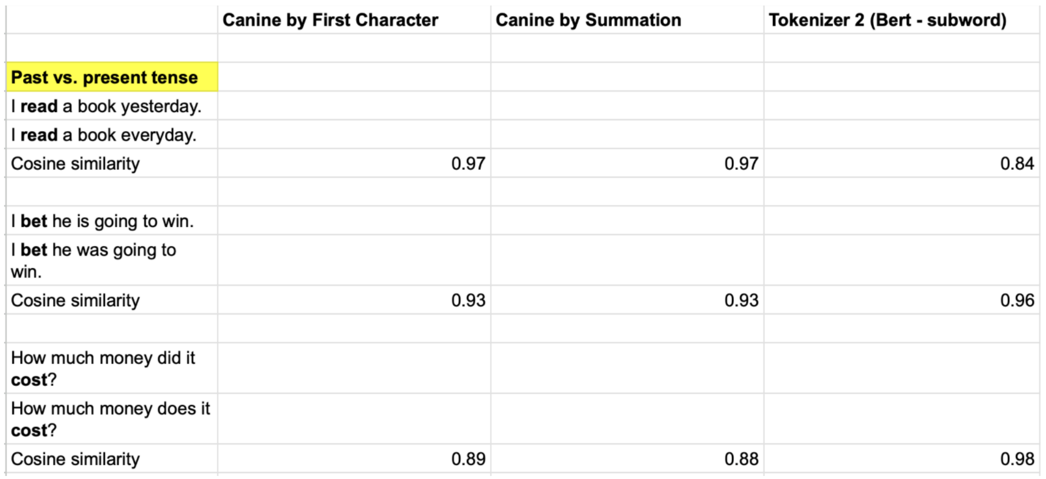
- **Synonyms**: 2 sentences are identical except one of the word is replaced by a synonym of the original word. Synonyms should have high cosine similarity even without context. Better performing model should have high, but not 1, cosine similarity because they are different words that have the almost identical meaning and contexts. This test should reveal the effects of different spellings and will serve as benchmark for misused words, which is when the meanings of the original word is abandoned.
  - For example, "My favorite season is **fall**." versus "My favorite season is **autumn**."
- **Misused words**: 2 sentences are identical except one of the word is replaced by a word with unrelated meaning. The misspelling we chose is a common word so that the BERT model does not break the misspelling into multiple tokens. We expect better performing models to produce higher cosine similarity if we supply more contexts.
  - For example, "**it's**" versus "**its**".
- **Homonyms** a word is used twice in the same sentence with two different meanings. The challenge for the homonyms is that the 2 words are in the same sentence, so they are provided with the same context information. Thus, we can test how the models contextualize information based on the order within the text.

- For example, ”Johnny **saw** a his dad using a **saw** to cut some wood.”

- Sentence with **clauses** add more contextual meanings to the object or event we are describing via relative pronouns such as “what”, “that”, “which”, and “who”. We tested how well the models pick up information from clauses.
  - For example, “Johnny ate some **apples**.” versus ”Johnny ate some **apples** that caused him stomachache.”

## 5.4 Tests Results

## 5.5 Past versus Present tense

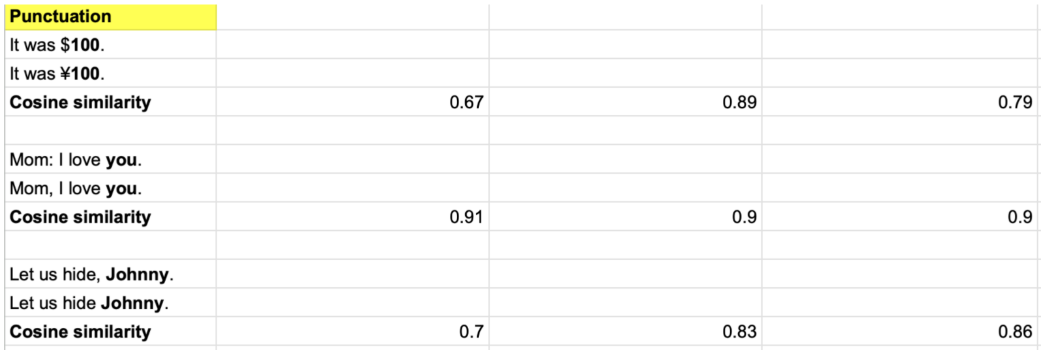


| | Canine by First Character | Canine by Summation | Tokenizer 2 (Bert - subword) |
|---|---|---|---|
| | | | |
| **Past vs. present tense** | | | |
| I **read** a book yesterday. | | | |
| I **read** a book everyday. | | | |
| Cosine similarity | 0.97 | 0.97 | 0.84 |
| | | | |
| I **bet** he is going to win. | | | |
| I **bet** he was going to win. | | | |
| Cosine similarity | 0.93 | 0.93 | 0.96 |
| | | | |
| How much money did it **cost**? | | | |
| How much money does it **cost**? | | | |
| Cosine similarity | 0.89 | 0.88 | 0.98 |

All three models have high and less than 1 cosine similarity between the past and the present tense. The three models appeared to be capable of embedding temporal meanings into the word without explicit spelling changes such as “do” vs “did” and “edit” vs “edited”.
One noticeable trend in BERT is that the cosine similarity is the lowest in the

first example, whose tense relies on the word “yesterday” and “everyday”. The other examples relied on the tenses of other verbs. Thus, BERT is most sensitive to to the direct meaning of other words, and less sensitive to more complicated tasks such as implying tense from the tenses of other words. The performance was opposite for both CANINE approaches. It is likely that the meanings of words do not affect CANINE. Moreover, both CANINE approaches produce similar results.

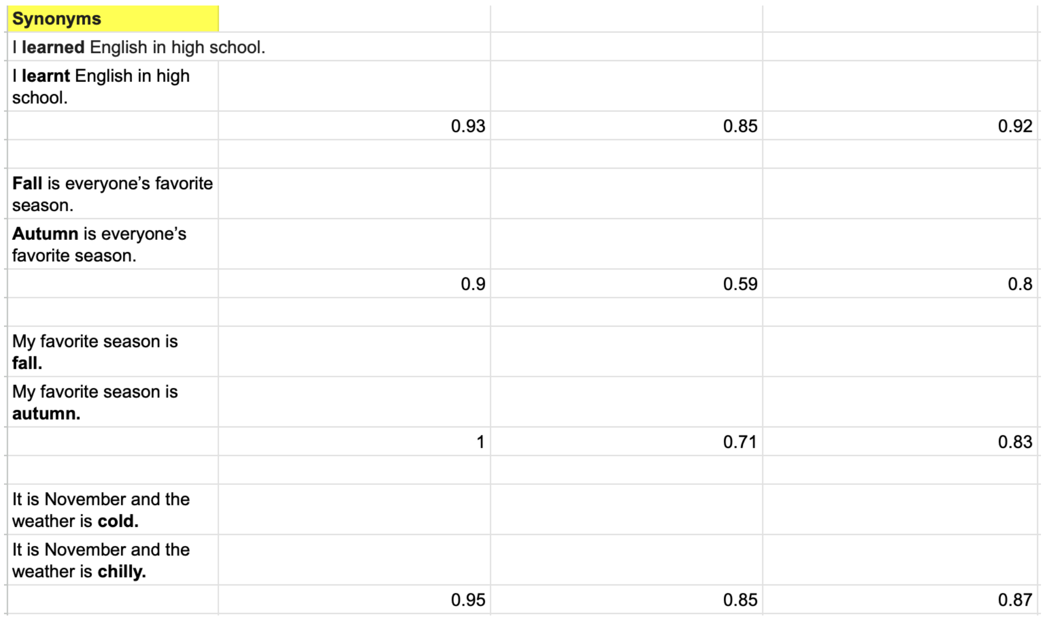
## 5.6 Punctuation



| **Punctuation** | | | |
|---|---|---|---|
| It was **$100**. | | | |
| It was **¥100**. | | | |
| **Cosine similarity** | 0.67 | 0.89 | 0.79 |
| | | | |
| Mom: I love **you**. | | | |
| Mom, I love **you**. | | | |
| **Cosine similarity** | 0.91 | 0.9 | 0.9 |
| | | | |
| Let us hide, **Johnny**. | | | |
| Let us hide **Johnny**. | | | |
| **Cosine similarity** | 0.7 | 0.83 | 0.86 |

All 3 models perform relatively well. CANINE first character provides the lowest cosine similarity. This shows that all 3 models can extract meanings from punctuation. The slight edge of the CANINE by first character suggests that tokenization by characters might be more effective than subword tokenization in interpreting punctuation because punctuation is single character.

## 5.7 Synonyms



| | | | |
|---|---|---|---|
| **Synonyms** | | | |
| I **learned** English in high school. | | | |
| I **learnt** English in high school. | | | |
| | 0.93 | 0.85 | 0.92 |
| | | | |
| **Fall** is everyone’s favorite season. | | | |
| **Autumn** is everyone’s favorite season. | | | |
| | 0.9 | 0.59 | 0.8 |
| | | | |
| My favorite season is **fall.** | | | |
| My favorite season is **autumn.** | | | |
| | 1 | 0.71 | 0.83 |
| | | | |
| It is November and the weather is **cold.** | | | |
| It is November and the weather is **chilly.** | | | |
| | 0.95 | 0.85 | 0.87 |

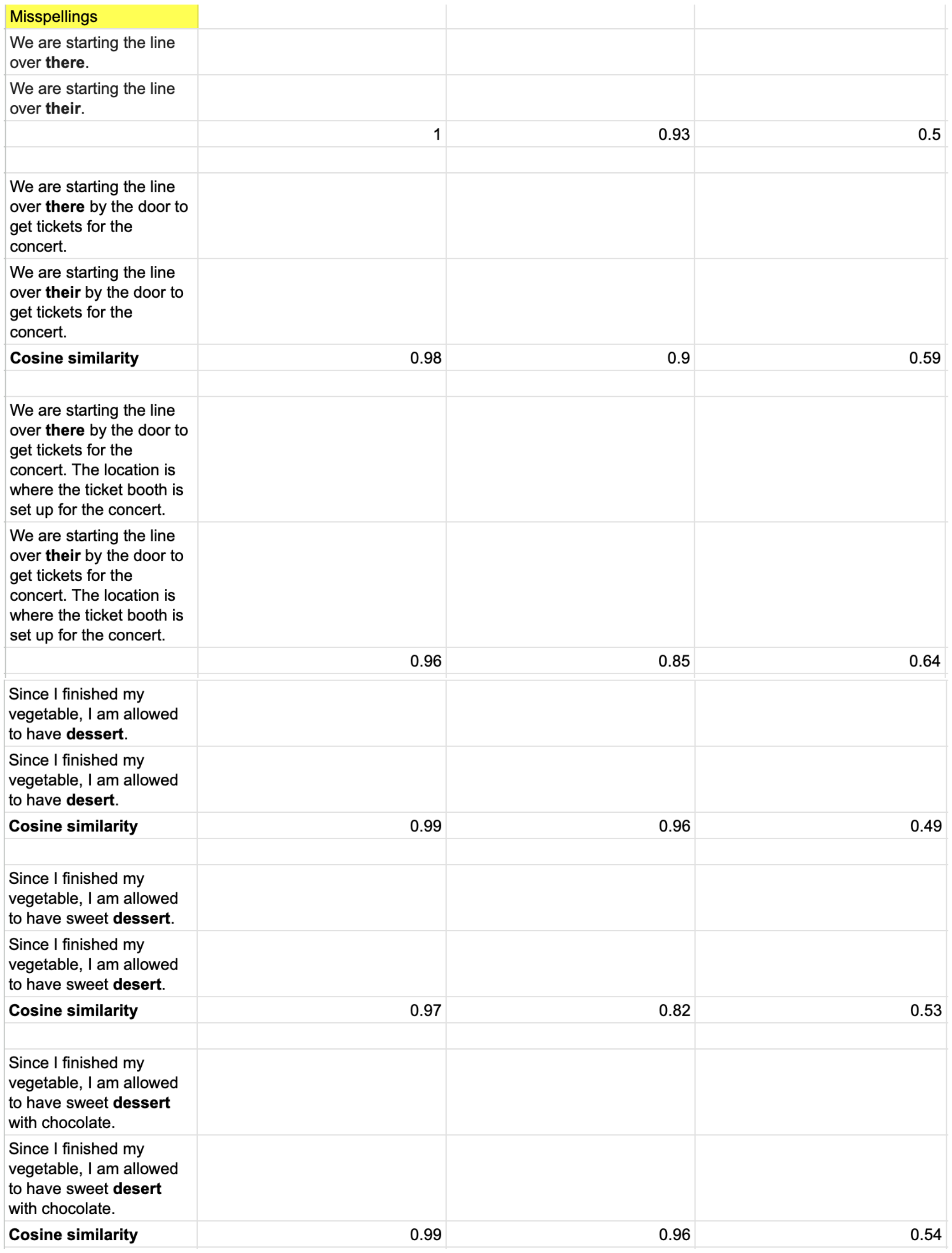
Most of the cosine similarity is high but lower than 1, which is sign of good models. However, there are 2 exceptions.

First, the first “fall” vs “autumn” example has very low cosine similarity in CANINE by summation. The next “fall” vs “autumn” example has much higher cosine similarity.

Second, the cosine similarity is 1 for the second “fall” vs ”autumn” example for CANINE by first character.

The two exceptions have no clear potential causes. They flag the unpredictability of the CANINE model in word embeddings.

## 5.8 Misused word



| | | | |
|---|---|---|---|
| Misspellings | | | |
| We are starting the line over **there**. | | | |
| We are starting the line over **their**. | | | |
| | 1 | 0.93 | 0.5 |
| | | | |
| We are starting the line over **there** by the door to get tickets for the concert. | | | |
| We are starting the line over **their** by the door to get tickets for the concert. | | | |
| **Cosine similarity** | 0.98 | 0.9 | 0.59 |
| | | | |
| We are starting the line over **there** by the door to get tickets for the concert. The location is where the ticket booth is set up for the concert. | | | |
| We are starting the line over **their** by the door to get tickets for the concert. The location is where the ticket booth is set up for the concert. | | | |
| | 0.96 | 0.85 | 0.64 |
| Since I finished my vegetable, I am allowed to have **dessert**. | | | |
| Since I finished my vegetable, I am allowed to have **desert**. | | | |
| **Cosine similarity** | 0.99 | 0.96 | 0.49 |
| | | | |
| Since I finished my vegetable, I am allowed to have sweet **dessert**. | | | |
| Since I finished my vegetable, I am allowed to have sweet **desert**. | | | |
| **Cosine similarity** | 0.97 | 0.82 | 0.53 |
| | | | |
| Since I finished my vegetable, I am allowed to have sweet **dessert** with chocolate. | | | |
| Since I finished my vegetable, I am allowed to have sweet **desert** with chocolate. | | | |
| **Cosine similarity** | 0.99 | 0.96 | 0.54 |

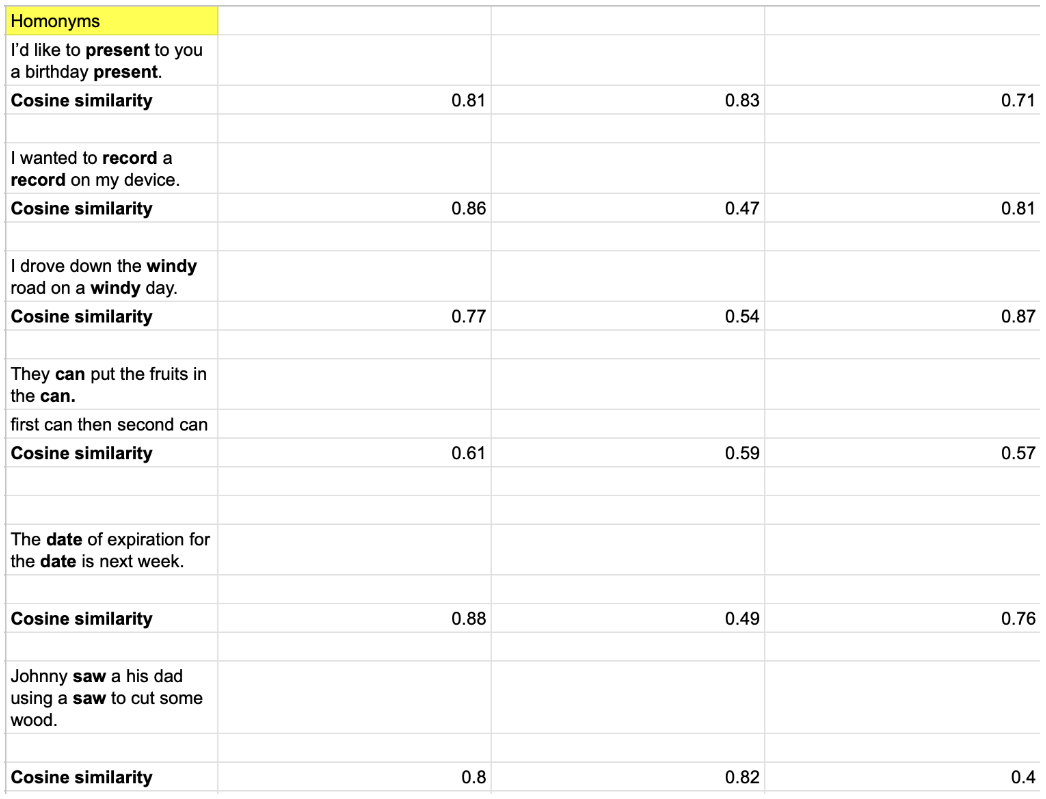
For BERT, the cosine similarity is around 0.5-0.6, even with more contextual information. This could be because the 2 pairs we originally use “there” vs “their” and “dessert” vs “desert” originally have very different meanings. Although BERT has the lowest cosine similarity, BERT still has excellent performance given that BERT uses a subword tokenizer. With more context provided, BERT gives higher cosine similarity, although not dramatic increase. Therefore, BERT is able to understand the original intention of the word, despite the misused word.

Both CANINE approaches have very high similarity, but the cosine similarity drops with more contextual information, which is contrary to our expectation.

One possible explanation is that the misused words have similar spellings. Because CANINE uses character tokenization, the 2 words have little difference in the first place. However, the decrease in cosine similarity with more contexts raises the doubts that perhaps CANINE processes context rather poorly. In addition, the CANINE by first character produces cosine similarity very close to 1. CANINE by summation produces cosine similarity closer to 0.9.

Thus, BERT is more sensitive to the original meanings of words, but is still able to absorb context enough to retrieve the intention of the speaker. CANINE is less affected by the original meanings of the words. However, its unpredictability raises questions about how each character embedding contains meanings.

## 5.9 Homonyms



| | | | |
|---|---|---|---|
| Homonyms | | | |
| I'd like to **present** to you a birthday **present**. | | | |
| **Cosine similarity** | 0.81 | 0.83 | 0.71 |
| | | | |
| I wanted to **record** a **record** on my device. | | | |
| **Cosine similarity** | 0.86 | 0.47 | 0.81 |
| | | | |
| I drove down the **windy** road on a **windy** day. | | | |
| **Cosine similarity** | 0.77 | 0.54 | 0.87 |
| | | | |
| They **can** put the fruits in the **can.** | | | |
| first can then second can | | | |
| **Cosine similarity** | 0.61 | 0.59 | 0.57 |
| | | | |
| | | | |
| The **date** of expiration for the **date** is next week. | | | |
| | | | |
| **Cosine similarity** | 0.88 | 0.49 | 0.76 |
| | | | |
| Johnny **saw** a his dad using a **saw** to cut some wood. | | | |
| | | | |
| **Cosine similarity** | 0.8 | 0.82 | 0.4 |

The cosine similarity produced by all 3 models are lower than those in synonyms in general. The cosine similarity is particularly low for CANINE by summation.

For BERT, this again illustrates its ability to embed meanings from other words and takes into account the location of words appeared in a the text.

CANINE by summation produces lower cosine similarity than CANINE by summation. This potentially can imply that the meaning of the entire words is distributed in more than the first character.

## 5.10 Clauses

| **Clauses** | | | |
|---|---|---|---|
| Johnny ate some bad **apples** that caused him to have a stomachache and he no longer wants to eat apples in the future. | | | |
| Johnny ate some bad **apples**. | | | |
| | 1 | 0.82 | 0.88 |
| | | | |
| They **eat** dinner. | | | |
| He **eats** dinner. | | | |
| | 0.83 | 0.73 | 0.35 |
| | | | |
| They **eat** a hearty dinner consisting of chicken soup and bread. | | | |
| He **eats** a hearty dinner consisting of chicken soup and bread. | | | |
| | 0.78 | 0.71 | 0.85 |

We used clause to add more sophisticated meanings to a word. The first example "apple" shows how a word with additional information compares to the original word. Both BERT and CANINE by summation have around 0.8-0.9 similarity. However, CANINE by first character has cosine similarity of 1.

The second and the third example should be viewed together. The verb "eat" in "they" and "eats" in "he" has only 0.35 cosine similarity in BERT without context. The cosine similarity dramatically increases to 0.85 when more information is provided to dinner–what is being eaten. For both approaches of CANINE, the cosine similarity is consistently around 0.8 and 0.7. This shows that BERT is much more sensitive to change in context.

## 6 Discussion of the Result

Based on the texts in the experiment, BERT shows consistent good performance in how it adds contextual information into its word embeddings. In summary, BERT is very sensitive to the meaning of others words in the sentence. For example, in the tenses test, BERT produces lower similarity when words like “yesterday” and “today” appear rather than inferring temporal meanings from other verbs. BERT also produces low cosine similarity for misused words, suggesting that BERT is very affected by the original meanings of the words. BERT’s sensitivity to the original meanings of the words can be explained by its subword tokenization. In the experiment, most of the words we tested are its own subword. 100 following the symbol yen is an exception.

CANINE by summation produces viable results most of the time. There is some instability observed. For example, the cosine similarity for the second synonym test is 0.59. CANINE has some advantages as well. For example, CANINE is less affected by the original meaning of the words, since it is trained on character level (“Canine”, n.d.).

CANINE by first character displays the worst result. It produces cosine similarity of 1 when 2 words have different meanings or different spellings. Its difference in the performance from CANINE by summation suggests the meanings of a word is not evenly distributed between the characters.

## 7 Conclusion

CANINE by summation is a potentially viable tool for producing context-dependent word embeddings. By adding up the vectors of each word, we managed to capture some meanings of the entire word. However, CANINE by first character is not a viable tool for producing such embeddings. BERT consistently produces good context dependent word embeddings throughout the experiment.

However, with all the shortcomings, CANINE performs better than we expected. Although CANINE is trained on character-level ("CANINE", n.d.), by summing up the vectors of each character, we are still able to retrieve some meaning of the word. As the authors who proposed CANINE said, CANINE does show its potential edge in flexibility (Clark et al., 2022). As we have seen in the misused example, CANINE is less affected by the original meanings of the texts.

The advantages and disadvantages of BERT and CANINE illustrate the influence of their tokenizers on word embeddings. Because BERT uses subword tokenization ("Summary of of the tokenizers", n.d.), BERT is more sensitive to word meanings. Because we used a character tokenizer for CANINE, CANINE is less sensitive to word meanings and distributes the meanings of a word to its characters unevenly.

## 8 Potential Areas for Future Research

One aspect of subword tokenization is that it can learn new words by combining known tokens (“Summary of the Tokenizers”, n.d.). In the future, we could test on how BERT tokenizes when the word we need to compare is broken down to two or more subwords. The challenge becomes how to compare the word embeddings of two words if each is made up of subwords. Perhaps we can sum up the vectors as in the CANINE by summation. We can assess if BERT’s performance maintains or drops with words made up of subwords.

As written by the author of CANINE paper, it is possible that CANINE performs better for languages where characters themselves carry significant meanings such as Chinese, Japanese, and Korean. We can repeat the experiment on a logogram languages using BERT and CANINE.