Exercise 2 Report: Building Word Embeddings with PyTorch

Michael Wagner, Tushardeo Manekar, Ivelin Ivanov

October 2023

Contents

1 Problem Statement

In this exercise, our task was to delve into the realm of natural language processing, specifically focusing on word embeddings. The main objective was to employ PyTorch, a powerful deep learning framework, to construct our own corpus-specific word embeddings. This report outlines the methods and strategies adopted, the datasets used, the model architectures chosen, the challenges faced, and the results obtained.

2 Explorative Data Analysis

2.1 Hotel Review Dataset

From the histogram (Figure 1), we can observe that a significant number of reviews have a length ranging between 0 to 2,000 characters, with a sharp peak around 500 characters. This indicates that the majority of the reviews are concise. As the review length increases, the frequency of such reviews decreases, showing a skewed distribution.

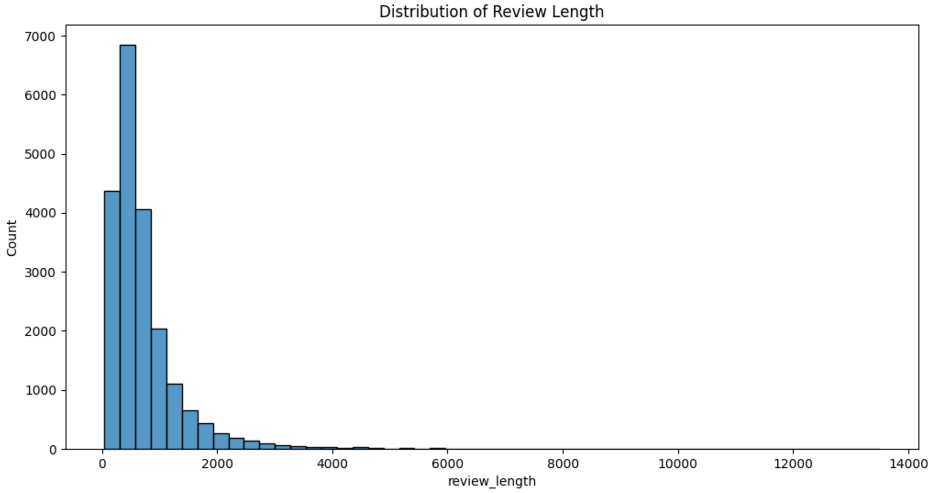

Figure 1: Distribution of Review Length

It's evident from the bar(Figure 2) chart that the majority of the reviews are positive, with a rating of 5 being the most prevalent. Ratings of 4 and 3 follow next in frequency, whereas ratings of 1 and 2 are comparatively less common.

These visual representations help us understand the overall makeup of the dataset, informing us that most reviews are short and positive in nature. Such insights are crucial for data preprocessing and model training, as they guide the decisions for handling outliers, data imbalance, and other potential challenges.

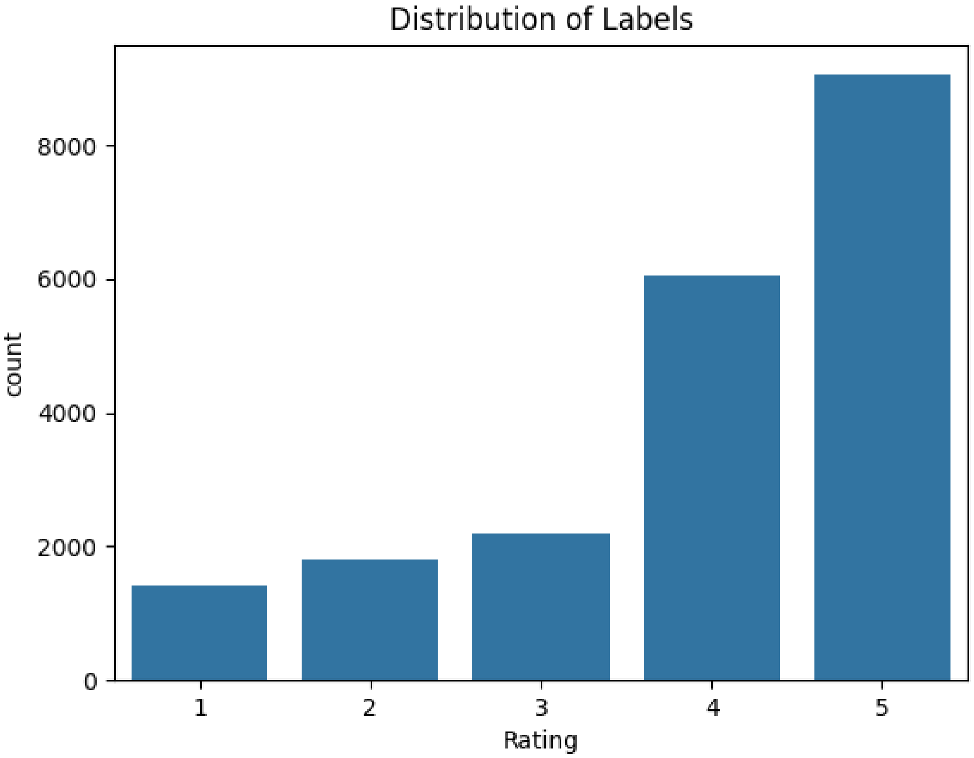

Figure 2: Distribution of Labels

```
        count   mean        std         min   25%     50%     75%      max
Rating
1       1421.0  769.534835  686.927121  77.0  351.00  564.0   943.00   6511.0
2       1793.0  867.002789  744.420751  74.0  433.00  653.0  1028.00   7802.0
3       2184.0  784.664835  743.864505  47.0  370.75  588.0   931.25  13501.0
4       6039.0  745.339957  729.579982  61.0  340.50  539.0   885.00  10062.0
5       9054.0  661.696488  627.032566  44.0  318.00  496.0   784.00  12738.0
```

2.2 Sci-Fi Dataset

The histogram (Figure 3) showcases the distribution of review lengths in the given dataset. The x-axis represents the length of the reviews, while the y-axis indicates the count of reviews for each length.

Several observations can be made from this visualization:

1. **Dominant Peak**: The most evident observation is the pronounced peak

at a review length of approximately 100-200 characters, suggesting that a majority of users tend to write reviews of this length.

2. **Steep Decline**: Beyond the 200-character mark, there is a rapid decrease in the number of reviews, indicating that longer reviews are less common.
3. **Sparse Long Reviews**: Reviews with a length of 600 characters and beyond are notably rare. This could imply that users rarely write extensive reviews, possibly due to the effort involved or platform limitations.
4. **Tail Distribution**: There are a few reviews that extend beyond 1000 characters, but these are exceptional cases and can be considered outliers.

In conclusion, the dataset mainly consists of shorter reviews, with a length of around 100-200 characters being the most common. It would be insightful to further investigate the content of these reviews to understand if shorter reviews are generally positive, negative, or neutral, and if the review length has any correlation with the sentiment of the review.

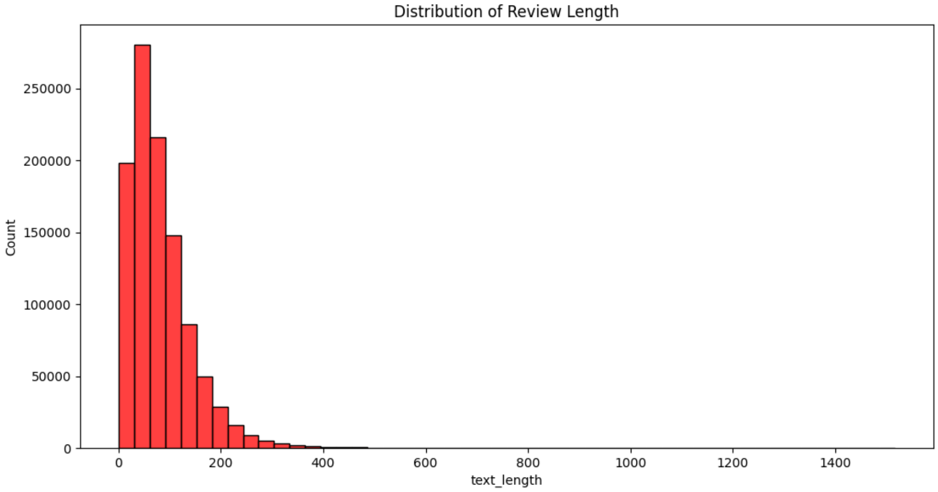

Figure 3: Distribution of Labels

```
count    1.044732e+06
mean     8.143872e+01
std      6.165343e+01
min      1.000000e+00
25%      3.800000e+01
50%      6.700000e+01
75%      1.080000e+02
```

```
max      1.516000e+03
Name: text_length, dtype: float64
```

3 Preprocessing Data & Model Architecture

3.1 Data Preprocessing: ‘preprocess_review‘ Function

The ‘preprocess_review‘ function is designed to take a review as input and perform a series of preprocessing steps to cleanse and normalize the review. The following are the individual steps involved:

1. **Lowercasing**: The entire review is converted to lowercase to ensure consistency and avoid case sensitivity during further analysis.
2. **Punctuation Removal**: All punctuation marks are stripped from the review, retaining only text-based words.
3. **Stop Words Removal**: Stop words are common words (e.g., ”and”, ”is”, ”the”) that are typically not important for semantic analysis in most contexts. These words are removed from the review.
4. **URLs and Web Links Removal**: Any web links starting with ”www.” or ending with ”.com”, as well as URLs, are removed since they are not relevant for the semantic review analysis.
5. **HTML Tags Removal**: All HTML tags are extracted and discarded as they don’t contribute to the actual content of the review.
6. **Numbers Removal**: Any numbers present in the review are removed to focus solely on the textual content.
7. **Emoji Conversion**: Emojis in the review are converted into their textual representation using ‘emoji.demojize‘.
8. **Mentions Removal**: Mentions, represented by ”@username”, are stripped off the review.
9. **Stemming**: The review is tokenized and each token (word) is stemmed to its root form. This helps in reducing the dimensionality of the data and consolidating similar words.

3.2 Model Architecture

The architecture for building word embeddings is based on the Continuous Bag of Words (CBOW) model, a popular neural network model used in Word2Vec for predicting the target word from a set of context words.

3.3 Network Structure

The model consists of multiple layers, as follows:

1. **Embedding Layer**: This layer is responsible for converting each word into a dense vector representation, capturing the semantic meaning of the word. The embedding layer uses a vocabulary size and an embedding dimension specified during model instantiation.

2. **Linear Layers**:

    - **First Linear Layer (‘linear1‘)**: The input to this layer consists of context words from both left and right of the target word. Each word is represented as a dense vector from the embedding layer. Hence, the total input dimension for this layer is context width $\times 2 \times Embeddingdimension.Thislayertransformstheinputintoahiddendimension.$

- **Intermediate Hidden Layers (‘linear2‘, ‘linear3‘, ‘linear4‘)**: These are additional layers to introduce non-linearity and allow the model to learn complex relationships.

- **Output Layer (‘linear5‘)**: The output layer is designed to predict the probability distribution of the central word over the entire vocabulary. Therefore, it has vocabsize units as its output.

3. **Activation Functions**: ReLU (Rectified Linear Unit) activation function is used after each linear layer except the output layer, introducing non-linearity into the model and helping it capture complex patterns.

4. **Softmax Layer**: Used at the output to convert the logits into probability distribution over the vocabulary.

3.4 Forward Pass

During the forward pass:

- The context words are first embedded into dense vectors using the embedding layer.

- These embeddings are then flattened and passed through the linear layers with ReLU activations in between.

- The output from the final linear layer is passed through a softmax to produce the probability distribution of the center word.

3.5 Training

For training the model:

- Negative Log Likelihood Loss (NLLLoss) is used as the loss function, given that the model outputs log probabilities.

- The model uses an optimizer (specified during training) to update its weights based on the computed gradients.
- A training loop is implemented with progress bars for visual feedback. During each epoch, the model learns by predicting the center word based on context words and adjusting its weights using backpropagation.

```
class CBOW(nn.Module):

    def __init__(self, vocab_size, EMBEDDING_DIM,
    CONTEXT_WIDTH, HIDDEN_DIM, word_to_ix):
        super(CBOW, self).__init__()

        self.word_to_ix = word_to_ix

        # Embedding layer to represent words in a dense form
        self.embeddings = nn.Embedding(vocab_size,
    EMBEDDING_DIM)

        # The input to this layer is CONTEXT_WIDTH words
    from the left, and CONTEXT_WIDTH from the right.
        # Each word is represented as an EMBEDDING_DIM sized
     vector.
        self.linear1 = nn.Linear(CONTEXT_WIDTH * 2 *
    EMBEDDING_DIM, HIDDEN_DIM)

        # An additional hidden layer for better
    representation
        self.linear2 = nn.Linear(HIDDEN_DIM, HIDDEN_DIM)
        self.linear3 = nn.Linear(HIDDEN_DIM, HIDDEN_DIM)
        self.linear4 = nn.Linear(HIDDEN_DIM, HIDDEN_DIM)

        # The output layer predicts the central word, so it
    has vocab_size units
        self.linear5 = nn.Linear(HIDDEN_DIM, vocab_size)

    def forward(self, context_idxs):
        # Embed the words
        embeds = self.embeddings(context_idxs)

        # Flatten the embeddings
        embeds = embeds.view(-1, embeds.size(1) * embeds.
    size(2))

        # Pass through the layers
        out = F.relu(self.linear1(embeds))
        out = F.relu(self.linear2(out))
        out = F.relu(self.linear3(out))
        out = F.relu(self.linear4(out))
        out = self.linear5(out)
        # Predict the log probabilities of the center word
        log_probs = F.log_softmax(out, dim=1)
        return log_probs
```

4 Results and Evaluation

4.1 Results Hotel Review Dataset CBOW2

The results from the CBOW2 model trained on the hotel review dataset are intriguing. Some of the closest words to the queried terms seem intuitively connected, while others appear unrelated at first glance.

For instance, for the query ”hotel,” words like ”property” and ”resort” are synonymous or closely related in the context of accommodations. However, terms like ”nonsstop” or ”altamont” are harder to directly relate without further contextual information. Similarly, for ”room,” while ”rom” seems to be a possible typographical error or short form, words like ”stationperfect” or ”dramat” are more challenging to decipher without additional context.

The term ”staff” returns words closely related to hotel personnel, such as ”employee” and ”receptionist,” which demonstrates the model's capability to capture semantic relationships within the domain.

Testing:

- **hotel:** properti, resort, westin, nonsstop, altamont
- **room:** stationperfect, unmemor, rom, popluar, dramat
- **staff:** employe, greet, concierg, receptionist, childreninf
- **beautiful:** pricey, nice, love, bathroomi, chioc
- **larg:** small, bathroomi, ammen, kept, adequ
- **pleasant:** nice, encount, wonder, courtiou, serious
- **decorat:** downsid, cofe, typic, improvestaff, seemsto
- **book:** arriv, stay, suggest, visit, went
- **check:** arriv, problem, hrperson, inform, mealther

4.2 Results Hotel Review Dataset CBOW5

Testing: The CBOW5 model trained on the hotel review dataset shows more relation specific near words. For example breakfast is associated with hotel so this is more context specific than the CBOW2 model. The same thing you see for the word book, where you see words like stay visit or ebook, which are more context based words for hotel reviews.

- **hotel:** properti, breakfasttot, citi, westin, say
- **room:** apart, modernis, expstay, bedroom, hotel
- **staff:** varsa, proprietor, owner, peopl, rosanna
- **beautiful:** hyattmarriotthilton, outstand, frenchth, sleepingand, laid
- **larg:** small, kept, purport, great, safe
- **pleasant:** allgirl, wonder, twoa, requestsr, reason
- **decorat:** goticoth, vondel, mannerit, heart, wasreason
- **book:** sivori, visit, stay, high, ebook
- **check:** thoughtfulaft, let, polynesian, tojust, talk

4.3 Results Sci-fi Dataset CBOW2

The CBOW2 model trained on the sci-fi dataset provides even more eclectic results. Given the nature of science fiction and the imaginative use of language, it's not surprising to find unconventional associations.

For the term "time," while words like "winter" might be conceptually linked to time in the broad sense, terms like "cooki" or "voliin" are more abstract in their connection. The term "planet" returns words that don't immediately seem relevant, such as "nitwit" or "puhli." This could be due to the narrative contexts in which these terms appear within the dataset.

Words associated with "old," like "circuit" or "ultrafax," suggest a technological context, potentially inferring futuristic or advanced technological settings common in sci-fi narratives.

Testing:

- **time:** dowm, voliin, cooki, nr, winter
- **planet:** haphazard, posabilit, nitwit, ote, puhli
- **slope:** dsked, gailydress, doubl, uppiti, supersecreci
- **old:** medivim, circuit, conjug, goof, ultrafax
- **long:** waterloo, centerpoint, musd, left, deme
- **rich:** bluish, protrud, viewer, transform, tamiskefl
- **said:** refashion, huh, dutyfre, stf, ummm
- **cant:** drastic, custodi, professioo, dumb, noroton
- **scare:** likclv, tember, descript, sam, barl

5 Conclusion

In conclusion, the comparison between the CBOW2 and CBOW5 models in processing both the Hotel Rating and Sci-fi datasets reveals several notable findings. CBOW2 uses a window of two words on each side of the target word, capturing a localised context, whereas CBOW5 considers a broader context with a window of five words on each side. This often results in more diverse semantic relationships being captured.

The differences in nearest neighbours between the sci-fi and hotel review datasets reflect their inherently different content, styles and languages. While the Sci-fi dataset contains imaginative and future-oriented concepts, the hotel evaluation dataset focuses on experiences and services. This explains why in a sci-fi context the word 'time' might concern complex narrative elements, whereas in the hotel context the word 'room' primarily describes physical attributes and experiences.

In assessing the quality of embeddings, CBOW2 shows more direct and conventional associations for the hotel rating dataset, although there are some anomalies. CBOW5 considers a wider range of relationships, sometimes resulting in less intuitive neighbours. For the Sci-fi dataset, CBOW2 produces some abstract associations, while CBOW5 occasionally produces results that are even harder to ve