Exercise 2 - To Embed or Not to Embed ...

> Building Word Embeddings with PyTorch

Part 1 - Train your CBOW embeddings for both datasets

> Describe your decisions (preprocessing, class structure) in the lab report.

Data Pre-Processing

We choose six data pre-processing steps that are useful and meaningful for the CBOW model. They are:

- Special Characters Cleaning Clean the data by removing special characters and punctuation
- Character Casing Lowercase the words.
- Tokenizing Words Tokenize the text to words for further data processing functions.
- Stop Word Removal Here we remove words that in English stop words list.
- Stemming Here we do stemming for the text, which reduces words to their root form. We choose PorterStemmer, which is a helpful algorithm. There may be some over-aggressive reductions, but overall there are more benifits to do stemming.
- Handling Rare Words and Out-of-Vocabulary Words Here we handle rare words and out of vocabulary (OOV) words. We use FreqDist to calculate frequences of all words. We use nltk.corpus.words.words() as our vocabulary to handle OOV. We choose words over minimum frequence (10 as default) and words in vocabulary. Since handling OOV words cost much time, we set it default to False, which will be more convenient for peer reviewers.

Sample Comparison

Hotel Dataset

Before pre-processing:

'fantastic service large hotel caters business corporates, serve provided better wife experienced- nothing short world.the room upgraded superior room overlooking harbour marina large window 50 feet length, anniversary bottle champagne sent chocolates compliments management, expensive did not regret moment choice hotel ... '

After pre-processing:

'fantast hotel cater better wife short world room superior room overlook harbour marina window length sent compliment regret moment hotel ...'

Scifi Dataset

Before pre-processing:

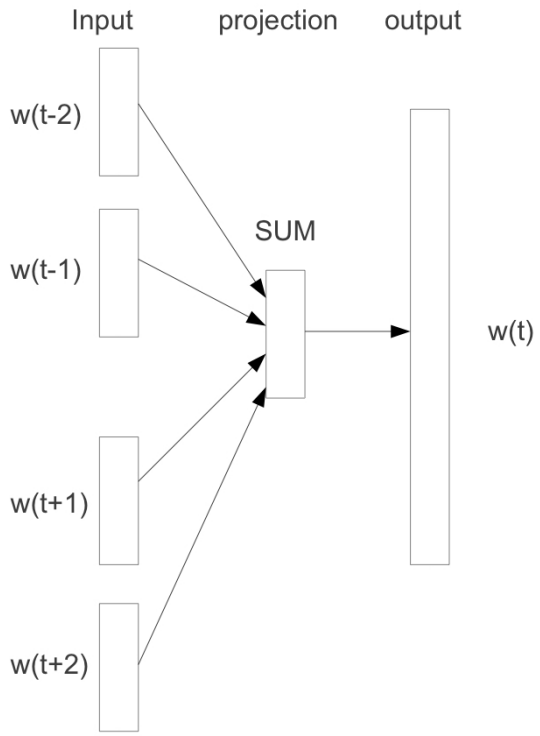
' A chat with the editor i # science fiction magazine called IF. The title was selected after much ...'

After pre-processing:

'chat editor scienc fiction magazin call titl select much ...'

Model Structure

Here is the screenshot from slides in class.



Source: [Mikolov 2013]

Here is our model structure code.

```
class CBOW(nn.Module):

    def __init__(self, vocab_size, embedding_dim, context_size):
        super(CBOW, self).__init__()
        self.embeddings = nn.Embedding(vocab_size, embedding_dim)
        self.linear = nn.Linear(embedding_dim, vocab_size)

    """
    Since the objective is to learn embeddings,
    instead of prediction or classification tasks,
    there is no need to add softmax layer.
    """
    def forward(self, inputs):
        embeds = self.embeddings(inputs)
        embeds = torch.sum(embeds, dim=1)
        out = self.linear(embeds)
        return out

# create your model and train.  here are some functions to help you make
# the data ready for use by your module
```

To be more specific, the model has:

- Input: A context, forming by a sequence of words, the number is determined by the context_size.
- Embedding Layer: The input words are mapped to their respective word embeddings using an embedding layer. These embeddings are learned during training. Embedding dimension is 50 in this task.
- Summation: The embeddings of the words in the context are summed up. This results in a single vector that represents the context.
- Linear: The summed context vector is then passed to a linear layer. It will map the context vector to a vector of the same dimension as vocabulary size.
- Output: The output can be used to predict the likelihood that words in the vocabulary being the center word. However, in CBOW, Since the primary objective is to learn embeddings, instead of prediction or classification tasks, there is no need to add a softmax layer.

> Are predictions made by the model sensitive towards the context size?

If we only focus on the losses the model reached, the answer seems to be no. After 20 epochs, both CBOW2 and CBOW5 model have losses around 5.8. However, seeing the 5 closest neighbours for the same 9 words, the results from CBOW2 are obvious better than those from CBOW5 (Specific results shown in Part 2). We think this is because a larger context window may consider more noise or unrelated words into the context. So the model is to some extend sensitive towards the context size.

Part 2 - Test your embeddings

> For the hotel reviews dataset, choose 3 nouns, 3 verbs, and 3 adjectives. Make sure that some nouns/verbs/adjectives occur frequently in the corpus and that others are rare. For each of the 9 chosen words, retrieve the 5 closest words according to your trained CBOW2 model. List them in your report and comment on the performance of your model: do the neighbours the model provides make sense? Discuss.

CBOW2_hotel

Chosen words with frequency:

nouns:

- 'hotel': 26584,
- 'staff': 8203,
- 'statement': 21,

verbs:

- 'eat': 1594,
- 'jump': 106,
- 'weigh': 16,

adjs:

- 'good': 8687,
- 'nice': 6533,
- 'calm': 104,

5 closest words (in order):

- 'hotel': 'impress', 'room', 'place', 'quit', 'time'
- 'staff': 'way', 'ruin', 'eye', 'wow', 'threw'
- 'statement': 'gon', 'motor', 'jean', 'pedestrian', 'oven'
- 'eat': 'sign', 'recommend', 'breath', 'studio', 'run'
- 'jump': 'cross', 'slot', 'dart', 'sent', 'teen'
- 'weigh': 'whirlpool', 'bite', 'shot', 'bland', 'swimsuit'
- 'good': 'great', 'excel', 'quit', 'fine', 'best'
- 'nice': 'great', 'excel', 'quit', 'love', 'good'
- 'calm': 'coconut', 'wont', 'deep', 'regular', 'w'

> Repeat what you did in 2. for the Sci-fi dataset.

CBOW2_scifi

Chosen words with frequency:

nouns:

- year: 8204
- bedroom: 284
- way: 11287

verbs:

- eat: 1193
- think: 10442
- look: 18611

adjs:

- good: 8129
- super: 286
- long: 8505

5 closest words (in order):

- 'year': 'extract', 'scan', 'allallu', 'day', 'entic'
- 'bedroom': 'deaden', 'instead', 'cari', 'rhythm', 'musingli'
- 'way': 'think', 'hing', 'alway', 'mean', 'want'
- 'eat': 'modern', 'opportun', 'thought', 'nausea', 'tang'
- 'think': 'know', 'sure', 'find', 'better', 'want'
- 'look': 'stare', 'turn', 'think', 'flew', 'mightier'
- 'good': 'fixtur', 'desert', 'hope', 'well', 'sure'
- 'super': 'shith', 'thirtieth', 'thish', 'titl', 'openli'
- 'long': 'cascad', 'elfor', 'pwf', 'first', 'fetzer'

We can observe that the results make sense. The retrieved closest words have the same part of speech and similar meanings. For example, the closest words of 'hotel' include 'room' and 'place', the closest words of 'good' include 'great'.

> How does the quality of the hotel review-based embeddings compare with the Sci-fi-based embeddings? Elaborate.

If just focus on our chosen words, the quality of the hotel review-based embeddings is better than the Sci-fi-based embeddings. For example, the model learns from the hotel review-based embeddings, the closet neighbor of 'good' is 'great', however, the closet neighbor of 'good' is 'fixtur' learned from the Sci-fi-based embeddings. Overall, chosen words from hotel reviews dataset have more meaningful neighbours than those from Sci-fi dataset. However, the Sci-fi-based embeddings are still useful. If we look at the word 'think', it has neighbours like 'know', 'find', 'want' which are similar active verbs.

> Choose 2 words and retrieve their 5 closest neighbours according to hotel review-based embeddings and the Sci-fi-based embeddings. Do they have different neighbours? If yes, can you reason why?

Hotel-based: 'good': 'great', 'excel', 'quit', 'fine', 'best' 'eat': 'sign', 'recommend', 'breath', 'studio', 'run'

SciFi-based 'good': 'fixtur', 'desert', 'hope', 'well', 'sure' 'eat': 'modern', 'opportun', 'thought', 'nausea', 'tang'

They have different neighbors. We think this is because contextual differences. The contexts of these words are different, then the words may have different embeddings. For example, for hotel reviews dataset, there are many similar words like 'good', 'nice', etc. for describing hotels, and they often appear together with nouns like hotels, rooms, so the model can learn better from the similar patterns. However, for the sci-fi dataset, it's a narrative article, the context may not be focused on assessing whether something is good or bad.

> What are the differences between CBOW2 and CBOW5? Can you "describe" them?

5 closest words (CBOW5):

- 'hotel': 'spring', 'grant', 'invest', 'concert', 'card'
- 'staff': 'coconut', 'palm', 'land', 'housekeep', 'face'
- 'statement': 'pour', 'al', 'nonetheless', 'towel', 'n'
- 'eat': 'spring', 'palm', 'overbook', 'facial', 'depress'
- 'jump': 'coconut', 'fever', 'hectic', 'wo', 'supper'
- 'weigh': 'freeway', 'nit', 'depress', 'lagoon', 'els'
- 'good': 'import', 'partial', 'gon', 'oyster', 'facial'
- 'nice': 'door', 'appoint', 'concert', 'wrong', 'partial'
- 'calm': 'builder', 'grant', 'cupboard', 'score', 'fell'

In CBOW2, the window size is 2, which considers the two words to the left and two words to the right of the target word, while CBOW5 considers five words before and after the target word

respectively. CBOW2 model learns the most often occurred-together words, while CBOW5 learns a wider range, which means that it considers more contexts of the document, and has a more comprehensive understanding of the document.

Seeing the 5 closest neighbours for the same 9 words, the results from CBOW2 are obvious better than those from CBOW5. We think this is because a larger context window may consider more noise or unrelated words into the context. With a larger context window size 5, the CBOW5 model has to consider a broader range of words. The dataset may be small or sparse, so the CBOW5 model may encounter infrequent or noisy words more often than CBOW2, making it harder to learn meaningful embeddings.