Assignment 1

by Xie zicheng

Code is available at: https://github.com/Zicheng2333/nlp_2024/tree/main/ass01

Q1

1 Language Modelling

Benchmark dataset: WikiText-103.

Model Input: A sequence of words or tokens.

Model Output: A probability distribution over the vocabulary of possible words or tokens in a language.

Evaluation Metric

- **Perplexity:** The exponentiation of the cross-entropy, normalized by the sequence length. A lower perplexity indicates that the model is better at predicting the next token.

$$\text{Perplexity} = \exp\left(\frac{1}{N}\sum_{i=1}^{N} -\log(p(w_i|w_1, w_2, \ldots, w_{i-1}))\right), \tag{1}$$

  where:

  - N is the length of the sequence.
  - w_i represents the actual token at position i.
  - $p(w_i|w_1, w_2, \ldots, w_{i-1})$ is the predicted probability of token w_i given the previous tokens.

Performance and Introduction of Top2 Models

- **The Retrieval-Enhanced Transformer (RETRO)** improves language models by integrating document chunks retrieved based on local similarity to preceding tokens. It utilizes a frozen BERT retriever to fetch relevant information, combined with a differentiable encoder and a chunked cross-attention mechanism. This architecture allows RETRO to process significantly more data during token prediction. RETRO achieves **a test perplexity of 2.4 on the WikiText-103 dataset**, indicating its effectiveness in language modeling.

- **Hybrid H3** enhances state space models (SSMs) by incorporating a new SSM layer, H3, designed to improve the capabilities of recalling earlier tokens and comparing tokens across sequences. This model achieves a **test perplexity of 10.6 on the WikiText-103 dataset**. This approach addresses the expressivity gap between SSMs and attention while delivering promising results in language modeling tasks.

2 Machine Translation

Benchmark dataset: WMT2014 English-German.

Model Input: The text or speech in the source language that needs to be translated.

Model Output: The translated text or speech in the target language.

Evaluation Metric

- **BLEU (Bilingual Evaluation Understudy):** Measures the overlap of n-grams (subsequences of words or tokens) between the generated translation and human-generated reference translations. A higher BLEU score indicates better performance.

$$\text{BLEU} = \text{BP} \cdot \exp\left(\sum_{n=1}^{N} w_n \cdot \log \text{P}_n\right), \tag{2}$$

  where:

  - N is the maximum length of the n-gram used (usually 4)
  - P_n is the accuracy of an n-gram
  - w_n is the weight(usually equal) of the n-gram
  - BP is the "penalty factor" that penalizes the difference between the length of the generated sentence and the length of the reference sentence.

Performance and Introduction of Top2 Models

- **Transformer Cycle (Rev)** relaxes the traditional parameter sharing used in Universal Transformers by implementing three strategies: Sequence, Cycle, and Cycle (rev), which assign parameters to each layer in a more efficient manner. The method proves effective when utilizing large training datasets, as evidenced by achieving a **BLEU score of 35.14 on the WMT2014 English-German dataset.**

- **Noisy back-translation** enhances neural machine translation by augmenting parallel training corpora with synthetic source sentences generated from back-translations of target language sentences. This research finds that back-translations obtained through sampling or noised beam outputs consistently yield the best results, particularly in resource-rich settings. The approach achieves **a BLEU score of 35 on the WMT'14 English-German dataset.**

3 Question Answering

Benchmark dataset: SQuAD1.1

Model Input

- **Question:** The user's question, typically represented as a sequence of words, tokens, or embeddings, depending on the model's architecture and tokenization strategy.

- **Context/Document:** A context or document that contains the information from which they must extract the answer. This context can be a paragraph, a passage from a larger document, or even a whole book.

Model Output: The primary output of a QA model is the answer to the user's question. This answer can be a short phrase, a sentence, or a numerical value, depending on the nature of the question.

Evaluation Metrics

- **Exact Match (EM):** A binary metric that assigns a score of 1 for a perfect match and 0 otherwise. (measures whether the model's generated answer matches the exact wording of the ground truth answer)

- **F1 Score:** A combination of precision and recall. A higher F1 score indicates a better match between the two.

$$F1 = \frac{2 \cdot \text{Precision} \cdot \text{recall}}{\text{Precision} + \text{Recall}}, \tag{3}$$

  where:

  - Precision is the number of correctly predicted tokens divided by the total number of tokens in the model's answer.

  - Recall is the number of correctly predicted tokens divided by the total number of tokens in the ground truth answer.

Performance and Introduction of Top2 Models

- **ANNA** integrates various approaches to enhance performance of pre-trained language models on question answering tasks. It introduces an extended pre-training task and a new neighbor-aware mechanism that emphasizes neighboring tokens, thereby capturing richer contextual information. It achieves an **F1 score of 95.719 and an EM score of 90.622 on the SQuAD 1.1 dataset**.

- **LUKE** introduces new pretrained contextualized representations for words and entities using a bidirectional transformer framework. It treats words and entities as independent tokens and employs a pretraining task based on BERT's masked language model, predicting randomly masked tokens in a large entity-annotated corpus from Wikipedia. LUKE also features an entity-aware self-attention mechanism that takes into account the types of tokens when calculating attention scores. It achieves **an EM score of 90.202 and an F1 score of 95.379 on SQuAD 1.1.**

Q2

4 Learn About Tokenizer

4.1 BPE(Bype Pair Encoder)

4.1.1 Principle of BPE

The core idea of BPE is to compress text by **substituting frequently occurring character pairs with unused characters**, gradually shortening the text. This iterative process, known as **encoding**, continues until there are no more frequent pairs in the text or all available characters have been used. During **decoding**, the process is reversed: adjacent subwords without a delimiter are concatenated directly, while a separator is inserted between subwords where a delimiter exists. If there are any subwords left that haven't been replaced but all tokens have been fully processed, they are substituted with a special token, such as `<unk>`. Figure 1 illustrates an example of how the BPE algorithm is applied.

Example:

Input: `ABABCDEBDEFABDEABC`

1. Most frequent pair = `AB`, replace with `G`:
   `GGCDEBDEFGDEGC`
   Substitution: `AB -> G`
2. Most frequent pair = `DE`, replace with `H`:
   `GGCHBHFGHGC`
   Substitution: `DE -> H`
3. Most frequent pair = `GC`, replace with `I`:
   `GIHBHFGHI`
   Substitution: `GC -> I`

Output: Compressed text: `GIHBHFGHI`, Substitution table: `AB -> G`, `DE -> H`, `GC -> I`

Figure 1. Byte Pair Encoding Data Compression Example

4.1.2 Advantages and Disadvantages

Advantages:

- **Efficient Dictionary Size Management**: BPE effectively balances the size of the vocabulary and the number of encoding steps. In practical applications, the corpus is often very large, making it impractical to include every token in the vocabulary. By merging frequent character pairs, BPE reduces the total number of tokens needed to encode the corpus,.
- **Adaptable to Various Languages**: BPE works particularly well with languages that use Latin-based scripts, where words are formed with prefixes and suffixes. Its ability to tokenize subwords makes it suitable for such languages.

Disadvantages:

- **Ambiguity in Subword Sequences**: For the same sentence, such as "Hello world", BPE may generate different subword sequences, which is illustrated in Table 1. These differences lead to different ID sequence representations, creating ambiguities that may not be resolved during decoding.

Subwords(_ means spaces)	Vocabulary id sequence
_Hell/o/_world	13586 137 255
_H/ello/_world	320 7363 255
_He/llo/_world	579 10115 255

Table 1. Ambiguity in Subword Sequences

- **Inability to Learn Random Distributions**: BPE's greedy algorithm is unable to learn from random distributions. While training on various subword sequences could improve the robustness of the model by allowing it to tolerate more noise, BPE does not handle this well due to its deterministic merging approach,.

4.2 BBPE

4.2.1 Principle of BBPE

BBPE extends the standard Byte Pair Encoding (BPE) algorithm by **operating at the byte level rather than the character level**. This allows it to represent any language's text in byte format, which provides advantages in compactness and avoids out-of-vocabulary tokens. BBPE is particularly useful for languages with rich character sets like Japanese and Chinese, where character-level models often result in large vocabularies.

4.2.2 Advantages and Disadvantages

Advantages:

- **No Out-of-Vocabulary (OOV) Issues**: BBPE eliminates OOV problems by converting sentences into byte-level sequences, ensuring that all characters, even rare ones, are represented using a fixed 256-byte set.
- **Compact Vocabulary**: BBPE operates at the byte level, creating a smaller and more efficient vocabulary that reduces the need for many rare character tokens, unlike traditional BPE.
- **Cross-Language Sharing**: BBPE naturally supports multilingual setups, as all languages' characters can be encoded as bytes, allowing vocabulary sharing across different languages without needing retraining.
- **Robustness to Noise**: BBPE can handle noise and non-standard characters better than BPE, as it includes all byte-level characters, making the system more robust.

Disadvantages:

- **Higher Computational Cost**: BBPE increases the length of tokenized sequences compared to character-level models, leading to increased computational costs during training and inference.
- **Lack of Contextual Semantics**: BBPE tokens may represent incomplete characters or arbitrary combinations, requiring additional contextual information to prevent ambiguity.
- **Invalid Byte Sequences**: While every character can be represented in valid UTF-8 bytes, BBPE might produce invalid sequences during decoding.

4.3 Unigram

4.3.1 Principle of Unigram

The unigram algorithm is a subword tokenization technique that builds a vocabulary based on the probability of each token in the corpus. Unlike BPE or BBPE, which merge frequent tokens, unigram assumes token independence. It begins with a large vocabulary and **prunes low-probability subwords** until reaching the target size. The loss function used is:

$$\text{Loss} = -\sum_{i=1}^{N} \log\left(\sum_{x \in S(x_i)} p(x)\right), \tag{4}$$

where:

- N is the number of words in the training set.
- $S(x_i)$ is the set of possible subword segmentations of word x_i.
- $p(x)$ represents the probability of a subword x.

In each iteration, **subwords that contribute the least to minimizing the loss are removed**. This trimming process typically eliminates 10-20% of the vocabulary. The unigram algorithm is language-agnostic and is often implemented with SentencePiece. Despite its advantages, it faces challenges such as computational cost due to probability calculations and context insensitivity because it treats tokens independently.

4.4 WordPiece

4.4.1 Principle of WordPiece

WordPiece is a subword tokenization algorithm developed by Google, widely used in models like BERT. It builds a vocabulary by iteratively **adding subwords that maximize the likelihood of the training corpus**.

Process

1. **Start with a small vocabulary**: The initial vocabulary consists of individual characters and special tokens.
2. **Subword addition**: Subwords are added iteratively. The objective is to maximize the likelihood $L(C)$ of the corpus C, calculated by:

$$L(C) = \prod_{i=1}^{N} p(w_i | \, w_1, w_2, ..., w_{i-1}), \tag{5}$$

   where:

   - $p(w_i | \, w_1, w_2, ..., w_{i-1})$ is the conditional probability of the subword w_i given the previous subwords.

3. **Greedy tokenization**: For each word, the algorithm greedily selects the longest matching subword from the vocabulary.

4.4.2 Advantages and Disadvantages

Advantages

- **Effective OOV Handling**: WordPiece mitigates out-of-vocabulary (OOV) issues by breaking down unknown words into known subwords, reducing tokenization errors.
- **Adaptability**: WordPiece adapts well to different languages and domains by learning subword patterns from the corpus.
- **Preserves Root Information**: By preserving relationships between word roots (e.g., “small,” “smaller,” “smallest"), WordPiece helps models better understand word structures.

Disadvantages

- **Greedy Tokenization**: WordPiece chooses the highest frequency subwords, which can lead to overly coarse segmentations and affect language understanding.
- **Computational Cost**: The iterative nature of WordPiece can be computationally expensive, requiring large-scale data processing.
- **Balancing Granularity**: Finding the right granularity for subword splits can be challenging, potentially leading to overly fine or coarse tokenization.

4.5 SentencePiece

4.5.1 Principle of SentencePiece

SentencePiece is a language-independent subword tokenizer and detokenizer developed by Google. Unlike traditional subword segmentation tools that rely on pre-tokenized input, SentencePiece can be trained directly from raw text. This approach allows it to be fully language-agnostic and end-to-end.

Process

1. **No Pre-tokenization Required**: SentencePiece **treats the entire corpus as a sequence of characters**, ignoring pre-existing word boundaries. This is particularly useful for languages without explicit word segmentation, such as Chinese, Japanese, and Korean.
2. **Subword Segmentation**: SentencePiece uses either BPE (Byte Pair Encoding) or the Unigram language model to learn subword units directly from the raw corpus.

3. **End-to-End Training**: By integrating the segmentation process with the neural model training, SentencePiece ensures that the model is trained in a completely end-to-end manner.

4.5.2 Advantages and Disadvantages

Advantages

- **Language-Agnostic**: Since it does not depend on predefined word boundaries, SentencePiece is effective across various languages, especially non-segmented ones like Japanese and Chinese.
- **Lossless Tokenization**: SentencePiece ensures that tokenization is reversible, meaning the original input text can be perfectly reconstructed from the tokenized output.
- **Efficient Vocabulary Management**: It allows users to specify a target vocabulary size, offering more flexibility in controlling model size and performance.

Disadvantages

- **Context Insensitivity**: Like other subword models, SentencePiece treats tokens independently, which can lead to suboptimal tokenization for certain tasks that require more context awareness.

5 Train a Tokenizer

See code Q2.2 Train a tokenizer.

6 Train a Word Embedding Model

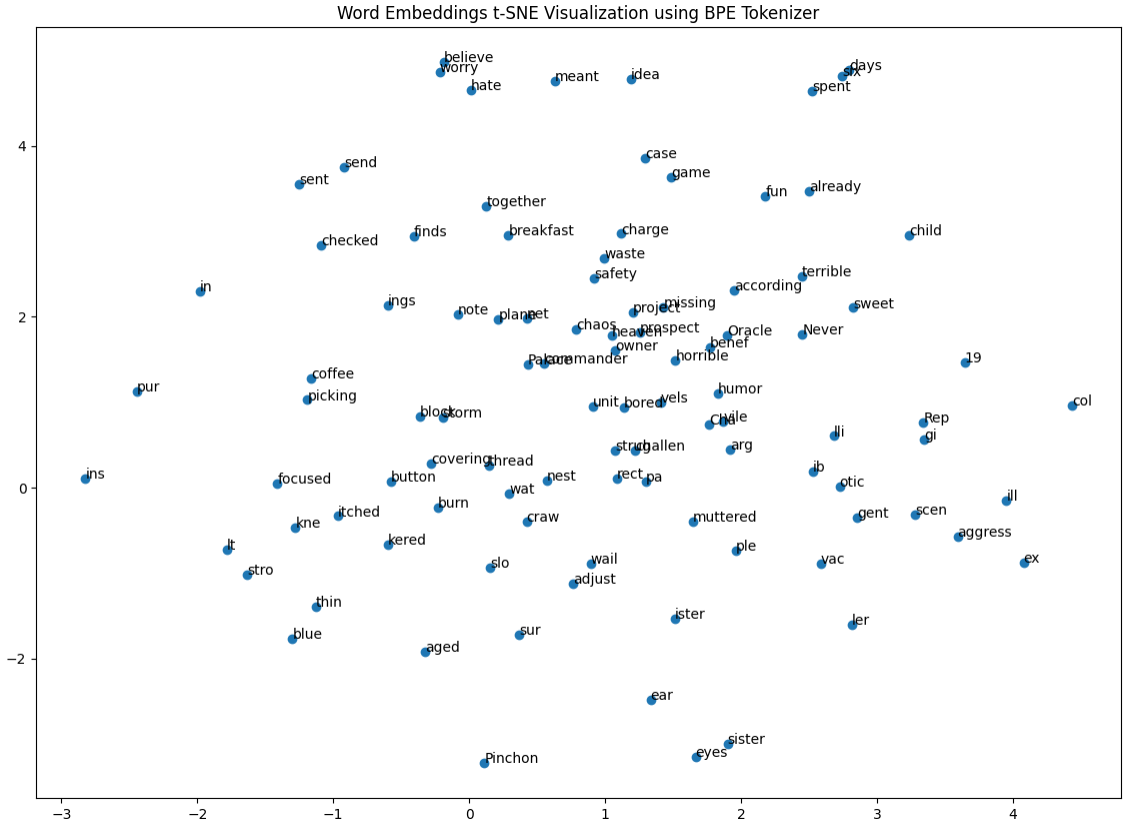

Figure 2. Word Embeddings Visualization Using BPE

Figure 2 shows the t-SNE visualization of 100 randomly selected tokens generated by the BPE

tokenizer and represented by the Word2Vec model. The original 100-dimensional embeddings were reduced to 2 dimensions for visualization using t-SNE.

7 More to Explore

7.1 BytePiece

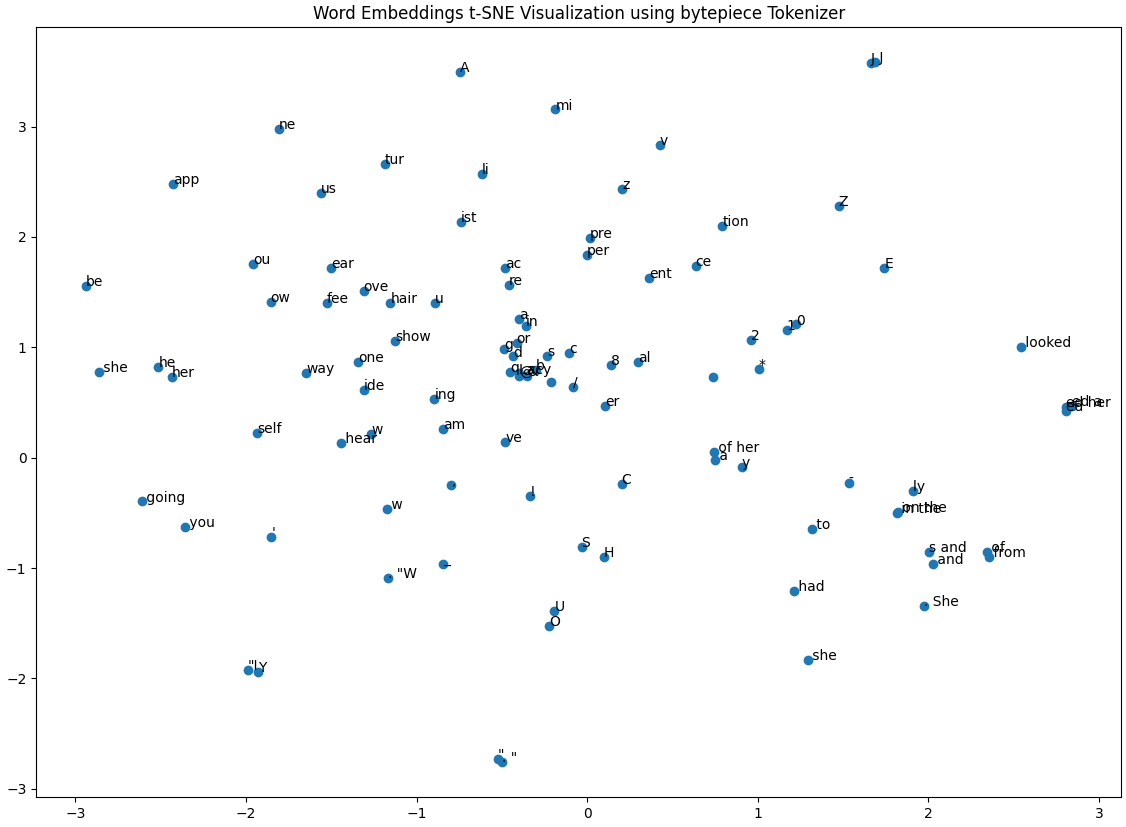

Figure 3. Word Embeddings Visualization Using BytePiece

Figure 3 shows the t-SNE visualization of 100 randomly selected tokens generated by the BytePiece tokenizer and represented by the Word2Vec model. The original 100-dimensional embeddings were reduced to 2 dimensions for visualization using t-SNE. BytePiece can also function as a tokenizer, positioning semantically similar words closer together.

7.2 Hand-written Implementation

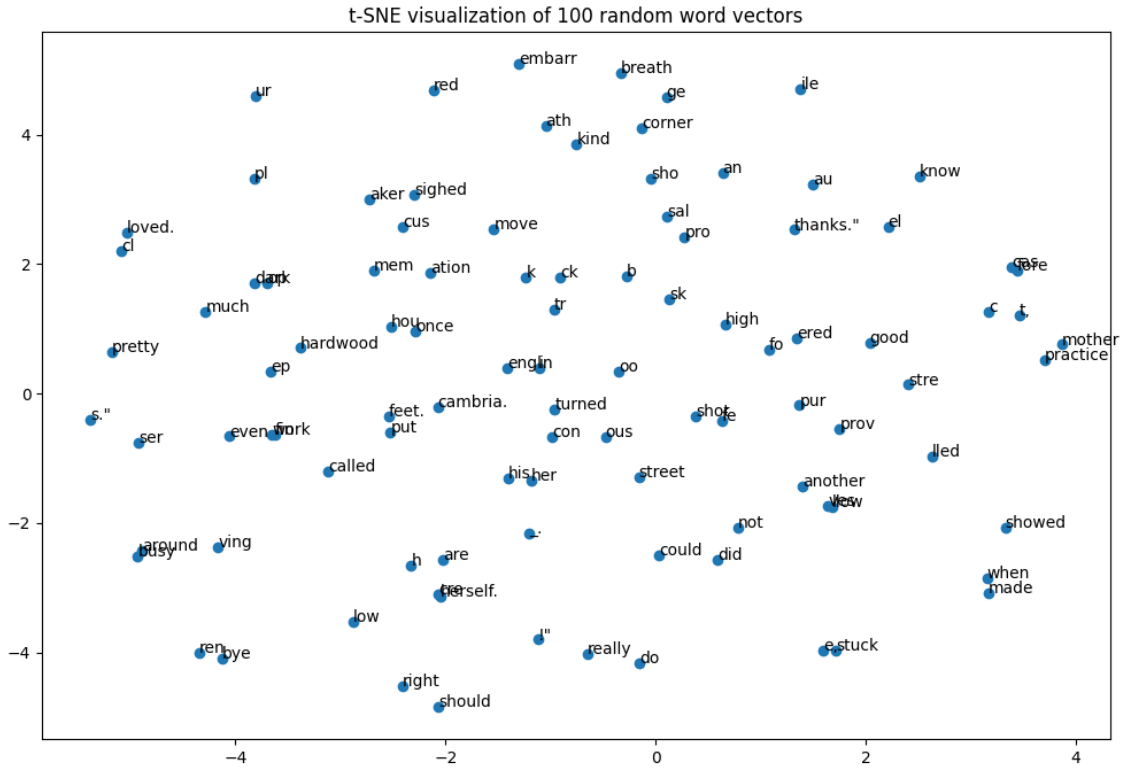

Figure 4. Hand-written Implementation

Figure 4 presents the t-SNE visualization of 100 randomly selected tokens generated by the custom BPE tokenizer and represented by the custom Word2Vec model. The original 100-dimensional embeddings were reduced to 2 dimensions for visualization using t-SNE. Although this custom model can function as an embedding model, it operates slower compared to publicly available toolkits.