© 2020 Xiaoyan Wang

HIBI: A HIERARCHICAL BIGRAM MODEL FOR ASSOCIATIVE LEARNING

BY

XIAOYAN WANG

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2020

Urbana, Illinois

Adviser:

Professor Chengxiang Zhai

ABSTRACT

There has been a shift of attention in the AI research where people gradually abandon traditional statistical models in favor of deep neural architectures. While effective in learning input-output mappings from two arbitrary distributions, the complex nature of neural models makes them hard to interpret.

In this thesis, we introduce a more interpretable hierarchical bigram (HiBi) model, which is extended based on the simple bigram language model. It contains a few components inspired by theories of human cognition, and has been shown through experiments to be effective in learning meaningful representation from sequential inputs without any labeling. We hope that HiBi could be a starting point to develop more complex cognitive models that are both interpretable and effective for representation learning.

To those who have supported me.

ACKNOWLEDGMENTS

I would like to thank Professor Chengxiang Zhai for his guidance and support, which have helped me greatly in deciding my research direction.

Human-like computational models have always been something that I wish to work on, yet I was hesitated to do so due to their relative lack of popularity. It was the few discussion with Professor Zhai that made me determine my mind, to work on the direction that truly motivates me, rather than any arbitrary direction that just "seems to work."

It has been an interesting experience working in this direction, mixed with a lot of excitement and frustration. It was truly satisfying to see pieces of knowledge from different domains that could be eventually put together to produce a meaningful output. Therefore, I am sincerely grateful for having Professor Zhai as my advisor, for this thesis would be impossible without him.

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION

1.1 MOTIVATION

Taken literally, the term “artificial intelligence” (AI) refers to building intelligence or consciousness by artificial (i.e. not nature) methods. By such a definition, the early appearance of AI could be found even before the common era. In Greek mythology, Talos was a giant automaton made of bronze that circled Crete three times daily to protect it from invaders [1]. There are also numerous occurrences of human-made characters in fiction literature that are capable of performing intelligent behaviors throughout history.

The dictionary definition of AI is the use of *computer* to simulate intelligent behaviors of humans [2], which began in the 1940s and 50s after the invention of the programmable digital computer. In summer 1956, the field of AI research was officially started during a workshop at Dartmouth College [3]. Since then, AI has gone through several alternations of booming and depressing periods.

Half of a century has passed, during which the form of AI gradually evolved. From simple cognitive programs that a human being is capable of computing by hand, the advancement of computational resources and the availability of a large amount of data nowadays has led to large-scale deep neural models that require days to compute even with super powerful machines [4, 5]. The promising results from modern AI research have attracted public attention and made many people believe that we are close to building intelligence that can ultimately replace people [6].

Even though there have been many AI systems that are able to achieve human-level performances under specific task settings (e.g., the ALBERT model [7] was able to achieve higher scores than human annotators on the Stanford Question Answering Dataset dataset [8]), we have yet to know whether those models truly have the cognitive capacity as humans do. Indeed, while the deep neural models are flexible and powerful at learning the input-output mapping between two arbitrary distributions, it is often hard to interpret their internal structure and to explain what they have learned that make them work.

Using a tool without knowing how it works is sometimes acceptable, especially if all we are interested in is to build a program that does the work. However, the lack of explanation limits the usefulness of those black-box machine learning models. Hence, the ultimate goal of studying AI is to continue pursuing the long-standing dream of building a model that resembles human minds, with the ability to learn, interact, and respond as we do. In this case, it would perhaps be necessary to understand the mechanics within the models, compare them with the cognitive theory of human minds, and discover the missing pieces that could bridge the two.

Motivated by this idea, we design the Hierarchical Bigram (HiBi) model, which is developed based on the traditional, statistical language model, and is extended to incorporate several key components inspired by research finding in the field of cognitive science. The goal of our work is to provide an interpretable algorithm for discovering implicit hierarchical structure from sequential data without supervision.

As a pioneer of AI research and a renowned cognitive scientist, Herbert Simon once criticize a "predicament" (as he called it) in psychology research that people often propose a new micro-theory to explain observations in each special experimental paradigm [9], and as he wrote:

> *"There exists a basic repertory of mechanisms and processes that Thinking Man uses in all the domains in which he exhibits intelligent behavior."*

We also see a similar trend in modern AI research, where distinct architectures are often used to tackle different variants of similar tasks. Therefore, it is our hope that the HiBi model we introduce here will be part of the "basic repertory" and the outputs could be reused by various kinds of downstream tasks.

1.2 OUTLINE

In Chapter 2, we will briefly introduce the n-grams language model, which is the foundation of our HiBi model. We will discuss a few prior works in Chapter 3, which were originally developed for different kinds of tasks, yet are related to our core idea. We will outline the similarities and differences between our work and theirs and explain what makes HiBi novel.

Chapter 4 and Chapter 5 are the central parts of the thesis, where we introduce the HiBi architecture and provide analysis on the experiment results. In Chapter 4, we will first outline the high-level flow of the model, then provide a detailed explanation for each of the internal components. For some components, there are a few possible implementations to choose from, as we will discuss in the context.

In Chapter 5, we will run our HiBi model on several publicly available datasets and provide the experiment results as references. We will provide interpretations for the model's output and perform analysis on the model's internal parameters after training it on unsegmented corpora.

We will then discuss the capacity and limitation of our model in Chapter 6 and draw a few analogies between our computational algorithm and theories of human minds. After that, we will suggest a few future directions that could potentially cover some of the drawbacks of the current version of HiBi.

Finally, we will conclude in Chapter 7 and revisit some of the key ideas that are mentioned earlier in the passage. We hope that as a simple model that has some properties cognitive process, the HiBi model could be a starting point for building a more elaborated model of human intelligence.

CHAPTER 2: BACKGROUND

In this chapter, we will first introduce the n-gram language models, which are the basis on which our HiBi model developed. We will discuss the reason we chose bigrams, which is a special case of n-gram where $n = 2$, as the basic unit at each level of our model.

2.1 N-GRAM LANGUAGE MODELS

A language model L is a function such that, when given a sentence with length m, computes the probability $P(w_1, \dots, w_m)$ of observing the sequence of words.

However, because a sentence can be arbitrarily long, in most cases, the possibility of observing any long sentence exactly as is in the corpus is close to zero, making the joint distribution over the entire sequence of words unsuitable for modeling natural language.

To relax the constraints, the n-gram language models were proposed, which assume that the probability distribution of observing each word in a sequence depends only on the $n-1$ words prior to it [10]. Therefore, the probability distribution of observing an entire sequence could be written as

$$P(w_1, \dots, w_m) = \prod_{i=1}^{m} P(w_i \mid w_{i-n+1}, \dots, w_{i-1}) \tag{2.1}$$

where

$$P(w_i \mid w_{i-n+1}, \dots, w_{i-1}) = \frac{\#(w_i \mid w_{i-n+1}, \dots, w_i)}{\#(w_i \mid w_{i-n+1}, \dots, w_{i-1})} \tag{2.2}$$

A few commonly used variants of n-gram language models include unigram, bigram, and trigram models, where $n = 1, 2,$ and 3, respectively.

As n increases, the model captures more contextual information, but the chances of encountering unknown n-gram also increase. On the other hand, smaller n, such as in unigram and bigram models, assumes too much independence and is bad at capturing long term dependencies.

Because higher n usually provides us with more meaningful results, people often choose a higher number for n whenever possible, and back off to smaller n when rare n-grams are encountered [11], that is, to approximate the n-gram distribution by:

$$P_{\text{bo}}(w_n \mid w_1 \dots w_{n-1}) \approx \begin{cases} P^*(w_n \mid w_1 \dots w_{n-1}) & \text{if } \#(w_1 \dots w_{n-1}) > k \\ \alpha_{w_1 \dots w_{n-1}} \cdot P_{\text{bo}}(w_n \mid w_2 \dots w_{n-1}) & \text{otherwise} \end{cases} \tag{2.3}$$

Another way to approximate the n-gram distribution is to take an interpolation of smaller n-grams:

$$P(w_n \mid w_1 w_2 \dots w_{n-1}) \approx \sum_{i=1}^{n} \lambda_i P(w_n \mid w_i \dots w_{n-1}) \tag{2.4}$$

where $\sum_i \lambda_i = 1$.

2.2 FROM FLAT BIGRAM TO BIGRAM HIERARCHY

Our HiBi model is designed to avoid the issue of computing the joint distribution while capturing dependencies across a relatively wide window, which is done by iteratively merging qualified bigrams into units and allowing them to be used as the elementary unit of bigrams for next iteration.

For the sake of clarity, we will use the term "token" to refer to an elementary unit at each iteration, which could either be the basic level inputs (e.g., a single character or word) or a bigram that was merged during the previous iteration, as illustrated in Figure 2.1.

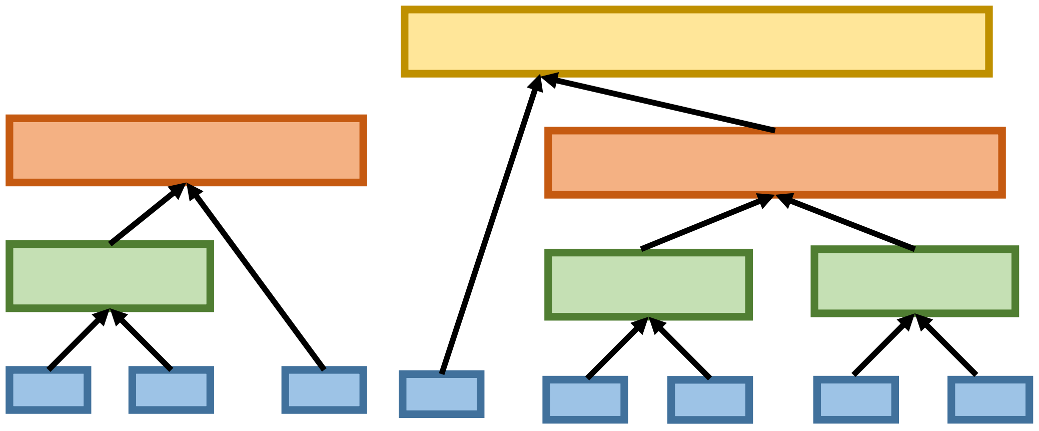

Figure 2.1: An illustration of the hierarchical bigram concept. At each iteration, the HiBi model parses the sequence and decides which of the bigrams should be merged. In the future iterations, the merged bigrams themselves would become candidates to form bigrams at the higher level, thereby constructing the "hierarchy."

Notice that only adjacent tokens are merged at each iteration. Since for each token t_i, we only consider t_{i-1} when we parse the sentences and perform the merging process, we consider this model as a hierarchical *bigram* model.

By iteratively constructing higher-level bigrams from meaningful units, our model effectively expands the context window when computing the distribution for the next token and have the same benefit of a longer n-gram. At the same time, because we merge only the high-score candidates that frequently appear together in the corpus, we avoid the potential issue of encountering rare n-grams.

In addition, the use of bigrams aligns well with the dependency grammar theory of languages, where sentences are also represented in tree structure [12]. Each level of the tree corresponds to a head-modifier dependency between words, which is a binary relationship in most cases. This motivates us to use only binary relations, which can still be powerful building blocks.

Another thing that is worth mentioning is that our HiBi architecture is unsupervised and requires no human labeling at all. The score of bigrams is determined purely by statistics of the tokens. Once new tokens are formed, they will be added to the same vocabulary as the tokens at the lower level.

From the model's perspective, there is no distinction between tokens at different levels. There is no weight assignment that encourage the model to use tokens at the higher level over those below them: we believe that, if a learned token represents a meaningful piece of information, then it should be a good predictor of the token adjacent to it. Therefore, any meaningful tokens should be naturally reused during sentence parsing and should be more likely to take part to form a higher-level token.

CHAPTER 3: RELATED WORKS

In this chapter, we will review a few models in other domains that are relevant to our work and explore their strengths and limitations, especially as models of the cognitive process. Notice that some of the models mentioned in this chapter are not designed to be cognitive models, and we do not mean to criticize them for lack of such an aspect. We are merely trying to examine the literature to what is yet to be done in building a model with interpretable intelligence backed by theories of minds.

3.1 HIDDEN MARKOV MODEL

The hidden Markov model (HMM) is a type of Markov chain that allows us to model the probability of unobserved events based on observations. A first-order HMM has chain-structure hidden states similar to a bigram model, as shown in Figure 3.1.

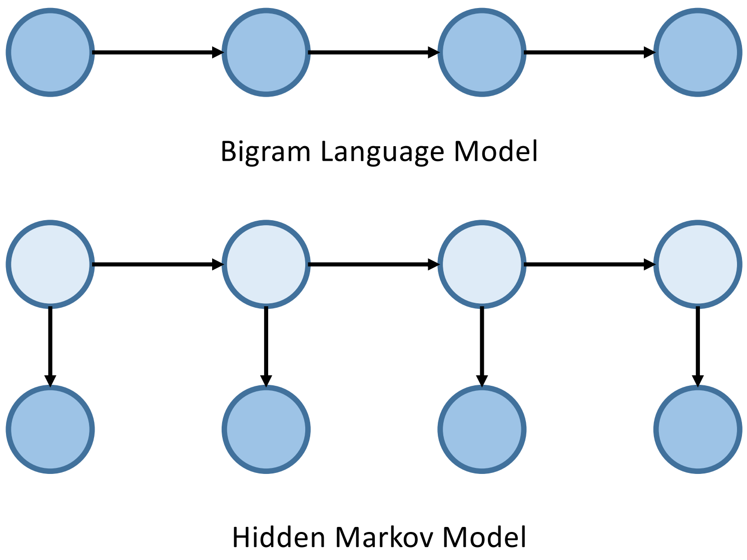

Figure 3.1: A comparison between a bigram language model and a first-order hidden Markov model, where the shaded nodes denote the observations.

In HMM, the probability distribution of a sentence is modeled by the joint probability over the hidden states and the emission probability from states to the observations. In natural language processing (NLP), HMM is often used for part-of-speech (POS) tagging, where each of the hidden states corresponds to a POS of the specific word that it links to.

If a corpus with POS labeling is available, then we could estimate the parameters for HMM just by counting the frequencies of word-POS pairs. Otherwise, we have to rely on the forward-backward algorithm to iteratively estimate the parameters [13].

Given a trained HMM and a sentence, we could efficiently find the sequence of POS tagging that maximize the probability of observing the sentence by using the Viterbi algorithm, which uses dynamic programming to store intermediate values to avoid repeat computation [14]. As we will discuss in Chapter 4, the Viterbi algorithm could also help us determine the sequence of tokens that lead to the optimal path during sentence segmentation.

A problem with the first-order hidden Markov model is that, similar to a bigram model, it assumes too much independence and is bad for capturing long term dependency. In addition, the number of possible hidden states has to be chosen in advance. A smaller number of states allow for better estimation of transition probabilities, yet having too few states could essentially convert the HMM into bag-of-words models. On the other hand, choosing a large number of hidden states could make the transition probability sparse. In the most extreme case where there is a one-to-one mapping between a hidden state and a word in the vocabulary, the HMM becomes equivalent to a bigram language model.

There are a few variants of the flat HMM, including the hierarchical hidden Markov model (HHMM), in which each state within the model could be another HHMM [15].

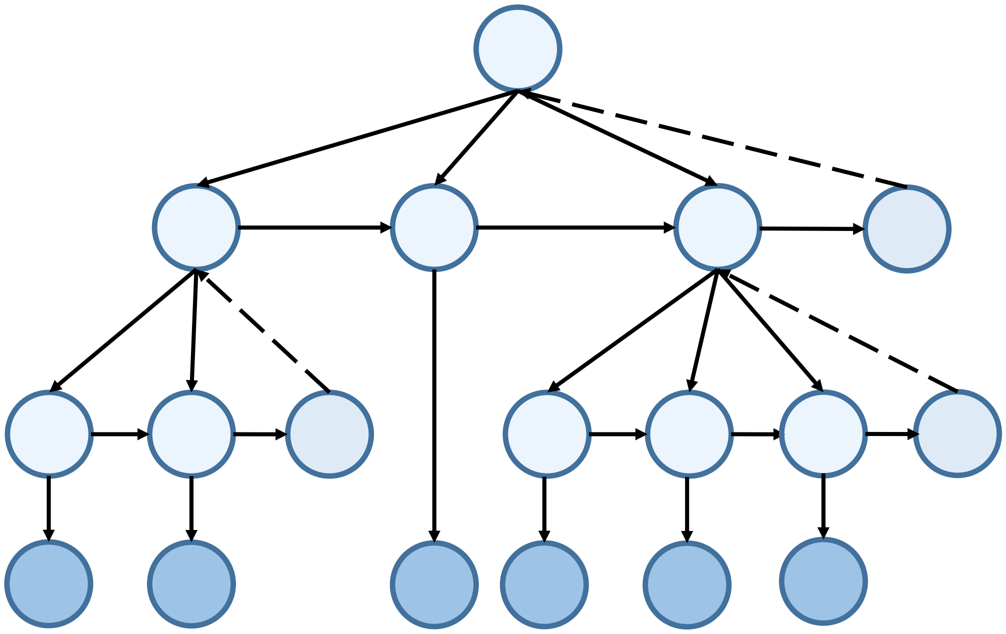

Figure 3.2: An illustration of the hierarchical hidden Markov model

Similar to a fully connected HMM, the HHMM enforces more constraints than the first-order HMM by not allowing arbitrary state transitions. At the same time, HHMM is more data-efficient than a fully connected HMM by making hierarchical assumptions on the sequence, as shown in Figure 3.2.

3.2 ELEMENTARY PERCEIVER AND MEMORIZER

The Elementary Perceiver and Memorizer (EPAM) was first introduced in [16] and was followed up by several revisions [17] as a model to simulate human route verbal learning. It was formulated with the following assumptions [18]:

1. The inputs are processed serially (rather than in parallel);
2. The system process the inputs in *chunks*, which is the largest component that the system is familiar with;
3. It takes time for a system to fixate on each chunk;
4. The system should be capable of holding memories of a few chunks temporarily;
5. The central processing mechanism fixate parts of the inputs that it attends.

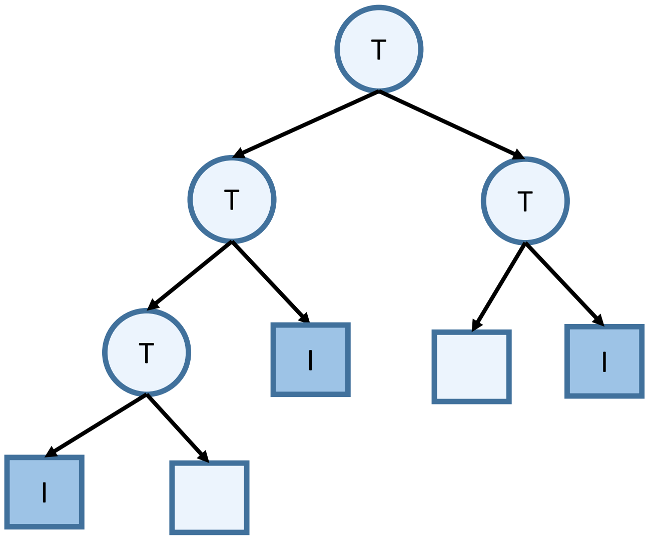

Figure 3.3: Illustration of a typical EPAM discrimination net, where T denotes the nodes that perform the discriminating test and I denotes the terminals with images associate with them.

As shown in Figure 3.3, a typical EPAM discrimination net has a structure similar to a decision tree. Every time the EPAM model perceives a new input, it recognizes the input through the discriminating testing nodes until it reaches a terminal. Then, it compares the difference between the image stored in the terminal with the inputs. A new discrimination node will be created to test the difference between the two, and the input, as well as the retrieved image, will be attached under the new testing node.

Our HiBi model is formulated under the similar assumptions as of the EPAM model, in that we also believe that a "chunk" (which is referred to as a token in our setting) is the elementary unit for processing. Our model also holds the memory of the previously seen chunk temporarily, even though in our case, only the last token is remembered when deciding the next token.

The difference between the EPAM model and our HiBi is that, in EPAM, the inputs are first remembered as a whole. As the model encounters more examples, it gradually breaks down the entire image into smaller features. We shall refer to this as the top-down approach. On the other hand, hour HiBi is a bottom-up approach, in which we iteratively grouping the elementary features together to form higher-level concepts.

The top-down approach has its benefits since in general, only a few examples of each class are required in order for it to learn the discriminations. However, for the same reason, it is more likely to be biased by unrepresentative features from the samples it has seen. On the other hand, the bottom-up approach requires more data to start with, but the features extracted are more likely to be statistically meaningful. As in humans' cognitive process, studies have suggested that both top-down and bottom-up mechanisms took place and interact with each other to process the incoming stimulus [19].

CHAPTER 4: HIBI, THE HIERARCHICAL BIGRAM MODEL

In this chapter, we are going to introduce the HiBi model, namely, an unsupervised **Hi**erarchical **Bi**gram model for learning tree-structure concepts from sequential inputs. The formulation of the model is written with the assumption that the inputs are natural language sentences, in which case the learned structural information would correspond to the lexical relationship between tokens. However, the model could also be easily extended to learn latent structures from any sequential data. The model is unsupervised, which means that the extraction of the structure relies purely on the statistical distribution of the data and no human annotation is necessary.

In Section 4.1, we will first give an overview of the HiBi model (Figure 4.1), introduce the few important components that the model consists of, and describe the algorithm for training HiBi. In Sections 4.2 to Section 4.5, we will dive into the individual component and discuss different implementations for each of them. In Chapter 5, we will evaluate the model on a few standard text corpus and provide quantitative and qualitative analysis.

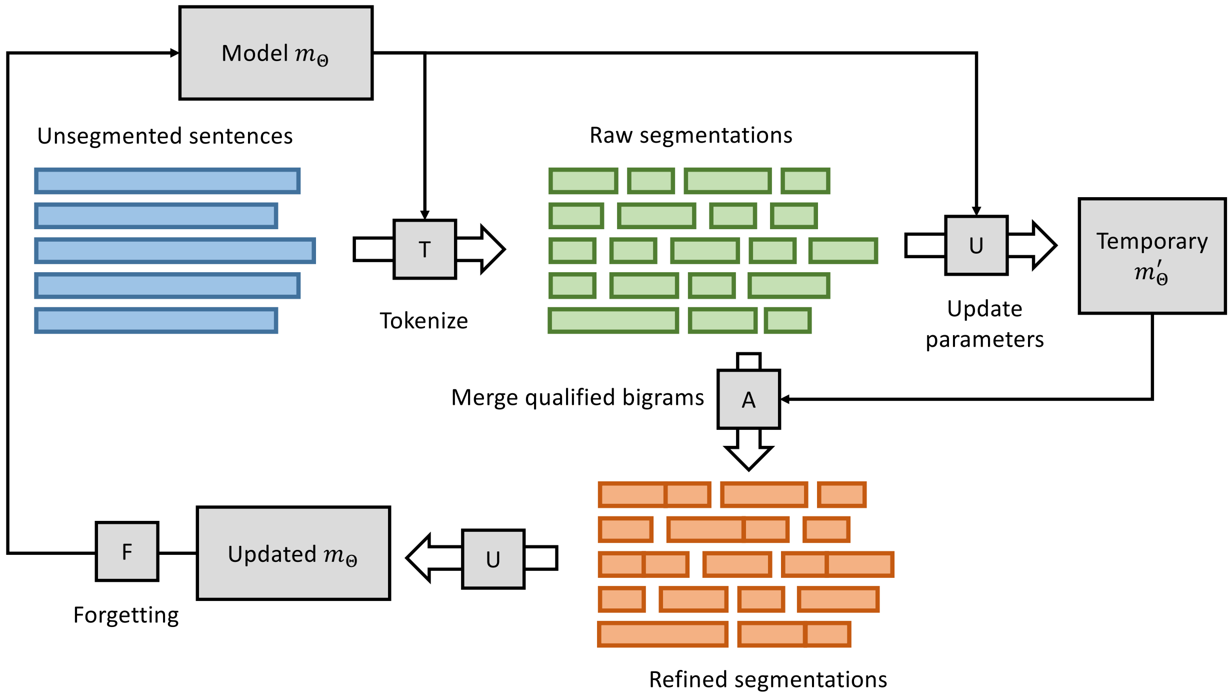

Figure 4.1: A high-level overview of the HiBi model showing the flow of data in a single iteration.

4.1 OVERVIEW

A HiBi model m consists of a set of learned parameters Θ as well as procedures T, U, A, and F that read from and update Θ as they handle the inputs. In the simplest version, Θ is responsible for maintaining the frequency of the bigrams that have encountered so far as well as a set of vocabulary V of learned tokens. We will discuss the meaning of the components in the later sections; but before that, the high-level ideas for each of the components are as follows:

- T (tokenizer): segments a sequence s of raw input in desired granularity (e.g., character-level, word-level, etc.) into a list of tokens $[t_1, \ldots, t_n]$, where $t_i \in V$.
- U (updater): updates the frequencies of the bigrams (t_a, t_b) stored in Θ based on a set of segmented sequences.
- A (assimilator): merges qualifying bigrams (t_a, t_b) to form a new token $t_{\text{new}} = t_a t_b$ and add it to V.
- F (forgetter): prunes the vocabulary V to remove rarely used tokens.

For simplicity, let (t_a, t_b) denotes a bigram pair such that t_a appears before t_b in the raw text. Let $\#(t_a, t_b)$ denotes the number of times that the model observes (t_a, t_b) from the input.

Given a corpus C consisting of sentences that have been pre-processed to the chosen granularity, the model will process the inputs and update the internal parameters using the following iterative algorithm (which is also illustrated in Figure 4.1):

1. Initialize a HiBi model m with an empty vocabulary.
2. Divide the corpus C into M batches, where M is a user-provided parameter.
3. For each $C_i \subseteq C$ where $i := 1, \ldots, M$
   (a) Segment each raw sentence s in the batch C_i into a set of tokenized sequences, $S := \{T(s; \Theta) \mid s \in C_i\}$
   (b) Update the parameters from the tokenized sequences S and stores them as *temporary* results, $\Theta' := U(S; \Theta)$
   (c) Using the updated parameters Θ', choose a subset of qualifying bigrams (t_a, t_b) to form new tokens $t_{\text{new}} = t_a t_b$ and replace the original occurrences with the new tokens, $S' := \{A(s; \Theta') \mid s \in S\}$ The new tokens and their original bigram structures should also be added to the vocabulary in this process.

(d) Adjust the parameters based on the refined segmentations and update the model $\Theta := U(S'; \Theta)$.

(e) Prune the vocabulary to keep the top L tokens that are most recently used.

At the end of the procedure, we will obtain a model with parameters $\Theta = (V, \#)$ where V gives us a list of learned tokens and $\#$ counts the frequencies of the bigrams consisting of the tokens in V.

In short, the model keeps grouping adjacent tokens that it is confident of treating as units, replace them in the original context, and continue the process iteratively. Since the newly formed tokens can also be part of a bigram in the next iteration, this algorithm allows us to learn a "bigram hierarchy," where the top levels correspond to phrases and lower levels correspond to words and morphemes.

In the following sections, we will discuss how each of the components should work and propose a few implementations. However, the components can also be replaced by any other existing models, as long as they follow the same input and output formats.

4.2 TOKENIZATION (T)

At this step, we are given a batch of raw sentences C_i and the current model parameter $\Theta = (V, \#)$, where V is the vocabulary of known tokens and $\#$ provides the frequency of the bigrams we have seen so far. Depending on the tasks, the basic unit of a sentence could either be characters, words, or even phrases. The goal for the tokenizer T is to utilize the learned parameters to segment the sentences into sequences of tokens.

Before going over the possible implementations of T, we shall note that, given the count of bigrams $\#$, we could compute the following probability distributions:

$$P(t_b \mid t_a) = \frac{\#(t_a, t_b)}{\sum_{t \in V} \#(t_a, t)} = \frac{\#(t_a, t_b)}{\#(t_a)} \tag{4.1}$$

if we use $\#(t_a)$ to denote $\sum_{t \in V} \#(t_a, t)$, which gives us the term frequency of t_a. [1]

$$P(t_a, t_b) = \frac{\#(t_a, t_b)}{\sum_{t \in V} \#(t)} = \frac{\#(t_a, t_b)}{N} \tag{4.2}$$

$$P(t) = \frac{\#(t)}{\sum_{t' \in V} \#(t')} = \frac{\#(t)}{N} \tag{4.3}$$

[1]Strictly speaking, $\sum_{t \in V} \#(t_a, t)$ only compute the number of times t_a appears *before* another token, so using this to replace term frequency can be problematic for tokens that usually appears at the end of sentences (e.g., punctuations). We could avoid the problem by appending a special end-of-sentence symbol after each sentence.

where N is the total number of tokens the model has encountered so far.

Because the tokens in the vocabulary have different lengths and some of them could be substrings of the others, in general, there will be more than one way to parse a sentence. Ideally, the model should choose the path to maximize its heuristic function.

4.2.1 Greedy Tokenizer

Given a sentence $s = [s_1, s_2, \ldots, s_l]$, where l is the length of the sentence, a greedy tokenizer selects the next token t by maximizing the probability of observing t given the last token. Let `<s>` and `</s>` denotes the special symbols used to mark the beginning and the end of sentences (which are not presented in the raw sentences). The greedy tokenizer run as follows:

```
1. Initialize $r := [$<s>$]$ to hold the list of segmented tokens (i.e., the results).
2. While $s \neq [\ ]$:
    (a) Initialize $c := \emptyset$ to hold a set of candidates
    (b) For $t \in V$:
        i. If $s[\ : |t|\ ] = t$ (i.e., prefix of $s$ exactly matches $t$):
            A. Add $t$ to the set of candidates $c$
    (c) If $c \neq \emptyset$ and $\#(r[-1], t) > 0$ for any $t \in c$:
        i. Choose the next token by maximizing the bigram probability $t := \text{argmax}_{t \in c} P(t \mid r[-1])$
    (d) Else, if $c \neq \emptyset$, which means no transition from previous token to any of the candidate has been observed so far:
        i. Use the fall back option to select the next token $t$ to maximize unigram probability $t := \text{argmax}_{t \in c} P(t)$
    (e) If $c = \emptyset$, meaning that the next basic unit of the $s$ has never been encountered:
        i. Set $t := s[0]$ and add $t$ to $V$.
    (f) Append $t$ to the end of tokenized sequence $r$
    (g) Updates $s$ by removing the parsed prefix, $s := s[\ |t| :\ ]$
```

At the end of the algorithm, we should obtain a segmented sequence $r = [t_1, t_2, \ldots, t_k]$ where $k \leq l$. Any unknown unit encountered during the process will also be added to the vocabulary.

The transition probability that is used in step 2(c) could be replaced by other scoring functions. For example, we could also select the next word by maximizing the pointwise mutual information (PMI)[20]:

$$t := \underset{t \in c}{\operatorname{argmax}} \ \operatorname{pmi}(r[-1]; t) = \underset{t \in c}{\operatorname{argmax}} \log \frac{P(r[-1], t)}{P(r[-1])P(t)} \tag{4.4}$$

In terms of computational complexity, the most expensive part of the algorithm is on 2(b), where the model loops over the entire vocabulary and checks the prefix. This could be optimized by storing the vocabulary in a prefix tree (i.e., a trie), which allows us to quickly retrieve all entries with a given prefix [21]. In fact, humans' mental lexicon might have a similar internal structure as a prefix tree, as it has been shown by psychological research that people are better at recalling words that *start* with a certain letter than words that have the letter in their third position [22].

4.2.2 Beam Search with Viterbi Algorithm

The greedy approach provides a simple and straightforward solution that can be quickly implemented. However, a problem with the greedy approach is that it does not account for the transition probability of future bigrams: a token that is optimal at the current might lead us to a dead-end and there is no way to go back.

If we think about how humans process the sentences while reading, it is likely that we are using a somewhat greedy approach. We segment a sentence whenever possible before reaching the end of it [23]. This also explains why people often interpret garden-path sentences incorrectly at first and have to correct ourselves later, which could happen when we read the following sentence [24]:

> *"The horse raced past the barn fell."*

To allow the model to consider more than one paths at a time when segmenting the sentences, we could expand the number of candidates that the HiBi chooses at each step. In order to guarantee that we could find the segmentation that maximize the joint probability of the entire sentence (based on bigram language model), we would have to explore *all* possible paths:

$$r := \underset{t_1, t_2, \ldots, t_k \in V}{\operatorname{argmax}} P(t_1 \mid \texttt{<s>}) P(t_2 \mid t_1) \cdots P(t_k \mid t_{k-1}) \tag{4.5}$$

which could required $O(|V|^l)$ computations in the wrost case and can be extremely expensive when V is large.

Since in bigram language model, the probability distribution of the suffixes depends only on the last state of the prefix, we could optimize Equation 4.5 using Viterbi algorithm, where, among all the paths that end with the same states, only the one with the highest score will be kept [14]. In addition, because humans' computational capacity is limited and we hardly ever consider all paths when we read sentences (yet we are able to comprehend most sentences well), instead of considering *all* possible paths, we could keep the top w paths that end with each state, effectively doing a beam search with width w [25]. An illustration of the Viterbi algorithm with a beam width of 3 is shown in Figure 4.2.

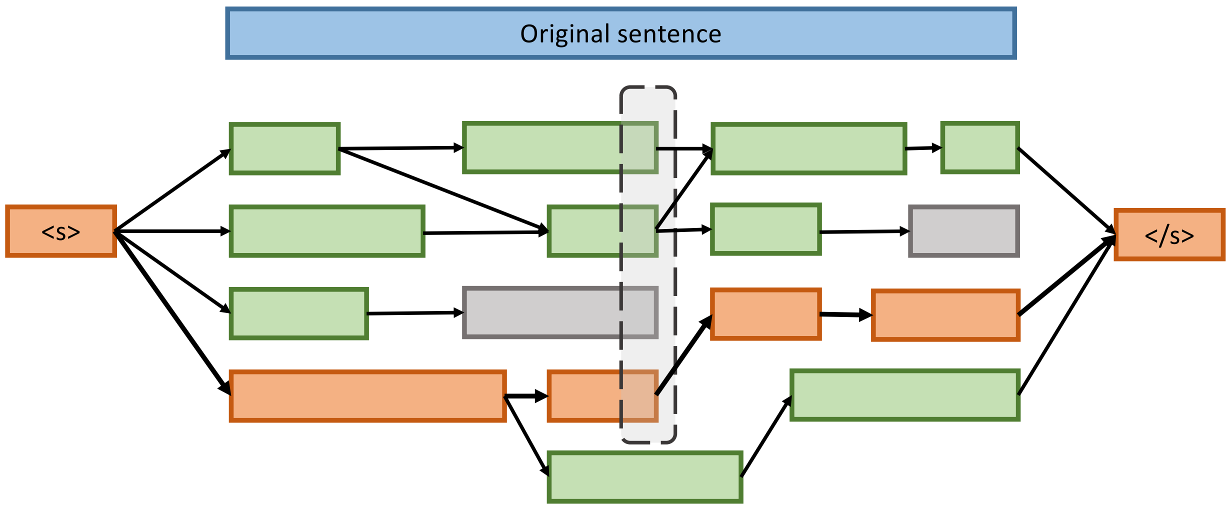

Figure 4.2: Illustration of the internal states of the Viterbi algorithm for sentence segmentation with a beam width of 3. The gray boxes denote states that have been pruned due to low score and the orange boxes denote the optimal path. Notice here that the beam width is applied when comparing states *end with* the same suffix, as highlighted by the dashed box.

Given a sentence $s = [s_1, s_2, \dots, s_l]$, the Viterbi algorithm with beam width of k proceeds as follows:

1. Initialize T as a array with length $l + 1$, where all entry of T are empty arrays.

2. Insert the `<s>`symbol as the start point of all paths $T[0] := (\texttt{<s>}, 0, -1)$, where the second and third entry corresponds to the score and the index of the source path (use -1 to denote the head of the path)

3. For $i := 1 \dots l$:

   (a) Prune the table to consider only the top k states ending with $s[i-1]$, $\mathit{states} := \text{prune}(T[i-1], k)$

```
   (b) Computes the set of tokens $t \in V$ that matches the prefix of the untokenized
       sequence, $c := \{t \mid t = s[\ i : i + |t|\ ]\}$
   (c) For each state $p_j \in states$:
       i. For each token $t' \in c$:
          A. Computes the next stage $next := (t', p_j.score + \log P(t', p_j.token), j)$
          B. If $T[i + |t'|]$ already has an entry $e$ for $t'$:
             • Replace the entry if $e.score < next.score$
          C. Else:
             • Append $next$ to $T[i + |t'|]$
4. Prune the final states, $f := \text{prune}(T[l-1], k)$
5. Initialize $r := [$</s>$]$, $loc := l - 1$
6. Select the state that gives the highest score (after adding the transition probability
   from the last token to </s>), $prev := \text{argmax}_j\, f[j].score + \log P($</s>$ \mid f[j].token)$
7. While $prev \neq -1$:
   (a) Append $T[loc][prev].token$ to the end of $r$
   (b) Updates $prev := T[loc][prev].from$ and $loc := loc - |T[loc][prev].token|$
8. Flip $r$ from front to back and return it as the segmentation.
```

4.3 PARAMETER UPDATES (U)

Given a segmented corpus S, the updater module counts the frequencies of the bigrams that appear in it and update the counts in Θ in addition to the existing values. Because we have added the `<s>` and `</s>` symbol at the beginning and the end of each segmented sentence, we are guarantee that $\#(t) = \sum_{t' \in V} \#(t, t') > 0$ for all t that appears in the corpus.

Another thing that is worth mentioning is that the updater is used twice in each iteration of the learning algorithm: once with the coarse segmentation corpus, and once with the refined segmentation corpus (which will be generated after merging qualified bigrams). In terms of implementation, there is no difference between the two, except that the updated parameters returns by the former procedure is only temporary and will be erased before the end of an iteration.

4.3.1 Memory Consolidation

As an optional step to increase the efficiency of the model as well as allowing the model to gradually adjust to the shift of token distribution, we could consolidate term frequencies that HiBi has accumulated so far into probabilities and reset the count to zeros. In this way, we avoid the problem of ever-increasing counts, which could overflow if the size of our corpus is huge enough.

Given a memory consolidation parameter $\alpha \in [0, 1]$ that controls the weight of information from previous iteration, the memory consolidation process simply go as follows:

$$P_i(t_a, t_b) = \alpha \cdot P_{i-1}(t_a, t_b) + (1 - \alpha) \cdot \frac{\#(t_a, t_b)}{N} \tag{4.6}$$

where P_i and P_{i-1} corresponds to the joint probability distributions of the bigrams for ith and $(i-1)$th iteration, respectively. A similar equation could be derived for the unigram probabilities and the conditional probabilities, where the updated distribution of each iteration is just a linear interpolation of previous iteration and the value calculated from the term frequencies of current iteration.

4.4 FORMATION OF NEW CONCEPTS (A)

The assimilator A is perhaps the most important component in the entire HiBi model. Given a segmented corpus S and a model m with parameter Θ, the assimilator is in charge of forming new tokens t_{new} by joining qualified bigrams (t_a, t_b) where $t_a, t_b \in V$.

The motivation behind such a mechanism is that, if t_a and t_b often occur together, then it is likely that $t_a t_b$ should be treated as an elementary unit. It is also inspired by the Hebbian theory on synaptic plasticity [26], that "neurons wire together if they fire together" [27].

Figure 4.3 illustrate the overall idea of the assimilator component. One thing to notice here is that the HiBi model only tries to merge bigrams at the level of the segmentation, and the merged tokens will not be part of another token until the next iteration. That is, given a segmented sequence s with length l, the refined segmentation s' returns by the assimilator would have length $l/2 \leq |s'| \leq l$.

Given a scoring function $a(t_a, t_b) \in \mathbb{R}$ that takes in two tokens $t_a, t_b \in V$ and returns a score, which should be proportional to the likelihood that they should be merged as a unit, and a threshold $\beta \in \mathbb{R}$ controlling the minimum score that is required in order for the bigram to be considered as qualified, the assimilator A works as follows:

1. For each segmented sequence $s \in S$:

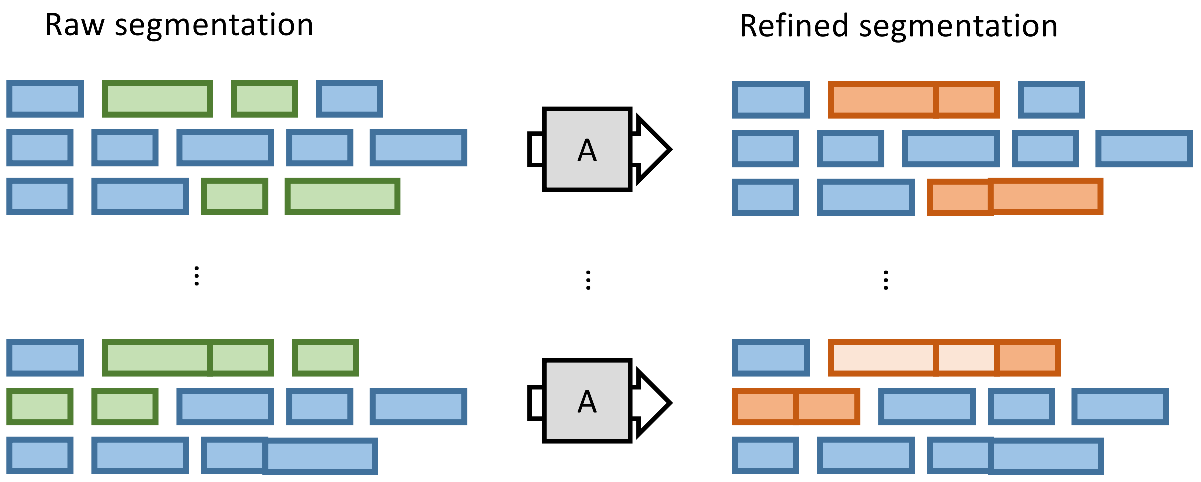

Figure 4.3: The expected inputs and outputs of the assimilator component A. In the first row, the assimilator replaced qualified bigrams consist of the elementary tokens with the newly formed units. The second row (which corresponds to a future iteration) shows that the learned units themselves could be part of bigrams and could be merged to form higher-level units.

(a) Computes the score between every pair of bigrams in s, $scores := [\ a(t_i, t_{i+1}) \mid i = 1, \ldots, |s|\ ]$

(b) While there exists any entry x in $scores$ such that $x > \beta$:

i. Select (t_i, t_{i+1}) such that $i := \text{argmax}_i\, scores[i]$

ii. Create a new token $t_{\text{new}} = t_i t_{i+1}$ and add it to V

iii. Replace the tokens (t_i, t_{i+1}) with t_{new}

iv. Set $scores[i]$, $scores[i-1]$, and $scores[i+1]$ to $-\infty$

2. Returns the refined segmentation with merged bigrams

In other words, for each sentence, the assimilator merges the qualified bigrams from highest score to lowest score (that still pass the threshold). The reason for this ordering, as well as the reason that adjacent cells are disabled in step 1(b)iv, is that it is possible to have a sequence t_a, t_b, t_c where both (t_a, t_b) and (t_b, t_c) are qualified bigrams. In this situation, we would like to choose to merge the pair with a higher score and mark the connection between the other pair as disqualified, to make sure that the model learns the most salient information at each iteration.

Some options for the scoring function a include the bigram transition probability or the normalized pointwise mutual information (NPMI) [28]:

$$a(t_a, t_b) := P(t_b \mid t_a) \tag{4.7}$$

$$a(t_a, t_b) := \text{npmi}(t_a; t_b) = \frac{\text{pmi}(t_a; t_b)}{h(t_a, t_b)} \tag{4.8}$$

where $h(t_a, t_b)$ is the joint self-information that could be estimated as $-\log P(t_a, t_b)$. The NPMI has an advantage over PMI in that the value of NPMI is in the range $[-1, 1]$ so that it is easier to set a meaningful threshold and to compare the scores across different bigrams. Other scoring functions that have similar properties could also be used, such as the significant score proposed by [29] that is calculated based on standard deviation and has been extensively used in phrase mining.

4.5 A VOCABULARY WITH LIMITED MEMORY (F)

After computing the refined segmentation, the vocabulary of the HiBi model will be augmented to includes the new words it has just learned. Then, the model will perform another round of parameter updates by going over the refined data. Whenever the model reads a token, it also moves the token to the top of the vocabulary list. As a result, before the forgetting step, the HiBi model will have an expanded vocabulary of tokens sorted by last retrieval time.

The forgetting step is rather straightforward: at the end of each iteration, F will prune the bottom of the vocabulary to keeps the top k tokens, where k is a user-defined number, which makes the vocabulary equivalent to a least recently used (LRU) cache [30]. When a token t is pruned, its term frequency will also be reset. In other words, if after several iterations, the same term t is added back to the vocabulary, it will be treated as a new token, and HiBi will start counting its frequency from 0.

There are several reasons that a forgetter F should be employed:

- Without the forgetting mechanism, the size of V will grow at each iteration, which will increase the computational load significantly, especially during the tokenization step (as the model has to consider all tokens that match the prefix of a sequence).

- The existence of more tokens in vocabulary also means that there the number of possible ways to tokenize will increase, which will increase the sparsity of the distribution and make it harder for the model to assemble new tokens from bigrams. In other words, it is better to keep a smaller vocabulary of necessary tokens and reuse them instead of keeping a large number of rarely-used tokens specific to the sequences.

- In some cases, the model might incorrectly group meaningless bigrams together and making incorrect associations. Because those tokens are not representative of the actual distribution of the bigrams, they will likely not be used for tokenization and will be pruned by the forgetting algorithm. That is, the forgetting algorithm F could also help to remove incorrect associations from the memory.
- Humans have limited memory and items that have not been rehearsed periodically will be forgotten [31]. Therefore, as a model that seeks to replicate humans' cognitive processes, the forgetting mechanism should also be included.

The reason to employ a forgetting mechanism based on last retrieval time instead of pruning the tokens with the lowest frequency is that

1. The newly added tokens will have lower counts than tokens that have been known for a long time, but we definitely do not want to remove them right after we add them to the vocabulary.
2. Tokens that are at the higher levels (i.e., longer tokens) tend to appear less frequently than tokens that are at the lower levels, but we still want to keep them, as long as they are still used from time to time. Ebbinghaus's forgetting curve actually suggests repetition as one way to improve the strength of memory [31].
3. Depending on the data, there could be a shift in the distribution of bigrams such that meaningful tokens that the model encounter during the early stage could be completely gone during the later stage. Therefore, it would be better to remove tokens that are not observed for a long period of time to better adapt to the new distribution.

Notice that if memory consolidation mechanism is introduced, in which case the probability distribution from the previous iteration will be discounted by a factor of α, then the forgetting mechanism could be simplified as one that removes the tokens with the lowest probability. Because tokens that are not observed during the current iteration will have discounted probability, they are more likely to be pruned.

After pruning the vocabulary via F, the algorithm reaches the end of an iteration and the updated model is ready to process the next batch of the corpus. The new tokens that are formed by merging elementary tokens will, in turn, become the elementary tokens for the next iteration. The process will continue iteratively until all inputs have been consumed.

CHAPTER 5: EXPERIMENTS

In this chapter, we will experiment HiBi on several publicly available datasets and report the outputs from our model. We will then provide analysis from the learned parameters and discuss what could we learn from the results.

5.1 DATASETS

Table 5.1 summarize the list of corpora we use for the experiments, all of which are available for download from the Natural Language Toolkit (NLTK) [32]. Among the datasets, the source texts of `Alice` and `Emma` are stories provided by Project Gutenberg [33], which is a library containing over $60,000$ free eBooks. `Brown` corresponds to the Brown Corpus, which contains 500 samples of English texts covering a wide range of topics [34]. The statistics shown in Table 5.1 were calculated based on NLTK's sentence and word segmentations.

Dataset	`Alice`	`Emma`	`Brown`
Number of sentences	1703	7752	57340
Number of unique words	2636	7344	49815
Avg. # of words per sentence	20.031	24.830	20.251
Avg. # of characters of per sentence	87.151	117.047	105.856

Table 5.1: Statistics of the corpora used for experiment

During training, we keep the sentence segmentations but discard the word-level ones. In other words, our HiBi model sees each sentence as a sequence of characters, but there is no special labeling about word boundaries. We convert all texts into lower cases while keeping the punctuation marks. As mentioned in Chapter 4, we insert special tokens for `<s>` and `</s>` to simplify probability calculation, but they should not affect the training results. From the model's perspective, there is no difference between regular English letters and spaces or punctuations. We would like to see if our HiBi model is able to successfully learn meaningful representations from raw texts without human intervention.

The following results are reported from three models trained from each of the corpora with hyper-parameters listed in Table 5.2. Normalized pointwise mutual information is used as the scoring function for bigrams across all experiments. After reaching the end of each corpus, we rewind the inputs and have HiBi parse it again from the beginning. We found that reviewing the corpus allows HiBi to form higher-level tokens based on the information it learned during previous iterations. For our experiments, this process is repeated 10 times for each corpus.

Dataset	`Alice`	`Emma`	`Brown`
Memory limit	3000	7000	30000
Batch size	100	100	300
Consolidation (α)	0.5	0.5	0.5
Merging threshold (β)	0.3	0.3	0.5

Table 5.2: Hyper-parameters used in experiments.

5.2 SENTENCE SEGMENTATION

Since all sentences are segmented into sequences of tokens before they are processed further, we are naturally interested in learning the effectiveness of the segmentation mechanism of our HiBi model.

Table 5.3 provides a few sample segmentation results from each of the corpora. All segmentation results are obtained using the Viterbi algorithm with a beam width of 3 (which is the same setting as in the training step).

`Brown`	after| |a| |while| , |we| |became| |aware| |that| |the money| |was| |disappear|ing |as| |fast| |as| |we| |re|plenished| |it .
	however , |she| |really| |does| |not| |know| |how to |match| |the |quantity| of |dollars| |given| |away| |by| |a| |quality| of |leadership| |that| |is| |bas|ically| |needed| .
	this is| |all| |the more| |remark|able| |because| |the |kir|o|v| |is| to |b|allet| |what| |senator| |gold|water| |is| to american| |politics| .
`Emma`	she| |had| |many| |acquaintance| |in the| |place| , |for| |her| |father| |was| |univers|ally| |ci|vil| , but| |not| |one| |among| |them| |who| |could be| |accepted| |in| |li|e|u| |of| |miss| |ta|ylor| |for| |even| |half| |a| |day .
	you| |do| |not| |think| |i| |could| |mean| |_you_ , |or| |suppose| |mr .| |knightley| |to| |mean| |_you_| .
`Alice`	would| |the| |fall| |never| |come| |to| |an| |end| !
	the| |jury| |all| |wrote| |down| |on| |their| |slates , ' |she| |doesn| |' t believe| |there ' s| |an| |atom| |of| |meaning| |in| |it| |,'| |but| |none of| |them| |at|tempted| |to| |explain| |the| |pap|er .

Table 5.3: Sample sentence segmentation results. The sentences displayed in the table are outputs of the HiBi model trained on the corresponding corpus shown in the left column.

As shown in the table, our HiBi model is able to discover the word boundaries for most of the time and even extract some common phrases (e.g., “this is” and “could be”). However, the model fails on rarer terms and still treat them as individual pieces.

Some interesting things happen with words that contain common affixes and prefixes. As shown in the sample output, the model decides to parse “disappearing” as “disappear” and “ing,” and does a similar thing with “remark|able,” “univers|ally,” and “re|plenished.” This actually coincides with how humans represent the words, and we expect the model to gradually merge the prefixes and suffixes and treat the words as a whole as it gets familiar with them.

5.3 TOP TOKENS IN VOCABULARY

Table 5.4 shows a side-by-side comparison between the top 20 tokens (measured by term frequency) on the `Brown` corpus. The last column shows the internal structure for each of the tokens.

Reference	Ours	Structure
the	‘ ’	‘ ’
,	a	a
.	s	s
of	in	(i, n)
and	,	(‘ ,’, ‘ ’)
to	.	(‘ ’, ‘.’)
a	was	(w, (a, s))
in	the	((t, h), e)
that	i	i
is	that	(t, h), (a, t))
was	he	(h, e)
he	it	(i, t)
for	as	(a, s)
“	for	(f, (o, r))
”	at	(a, t)
it	you	((y, o), u)
with	on	(o, n)
as	is	(i, s)
his	be	(b, e)
on	with	(w, ((i, t), h))

Table 5.4: Top 20 words with the highest frequency from the `Brown` corpus. The reference data is computed based on word segmentation provided by NLTK.

Since we treat spaces in the same way as other tokens, as a separator, it naturally appears the most frequent tokens among all. What is interesting is the amount of overlap between our learned token frequencies and the reference ones, and it shows that at least for the most common words, our HiBi model can correctly recognize them from the texts.

5.4 MINING LEXICON RELATIONSHIP BETWEEN TOKENS

To discover the lexicon relationship between tokens, we construct the adjacency graph from the segmented `Brown` corpus. We use the random walk approach as described in [35], but instead of taking the average on the paths of different lengths, here we only computing the scores by taking one step forward and one step backward (and does the same in the reverse direction). In other words, for a given keyword t, we score the tokens of other tokens $t' \in V$ by

$$s(t' \mid t) = \sum_u P(t \to u) \cdot P(t' \leftarrow u) \cdot \sum_v P(v \leftarrow t) \cdot P(v \to t') \tag{5.1}$$

where u and v are nodes in the adjacency graph.

Table 5.5 shows the example retrieval results with the keywords given on the top row of each column, where the retrieved tokens are ranked in descending order. We construct the graph on `Brown` corpus segmented by our HiBi model, with the same set of hyper-parameters as reported earlier. We connect each token to n tokens that appear directly before and after it, with $n = 6$ in our sample outputs (we have to use a larger window in this case, because our tokens could be elementary units smaller than a word).

september	**man**	**increase**
september	man	increase
june	s	change
july	it	rise
december	he	increases
october	one	changes
s	a	as
that	woman	a
chapter	is	increase of something
prior to june	i	year

Table 5.5: Example of top tokens retrieved by the given keywords.

We will have to confess that, currently, the tokens retrieved by this random walk approach are still very noisy. This happens because our segmented texts contain tokens of different granularity. The problem might be resolved by building adjacency graphs of variable window length based on some scoring measures, but we will leave that to future investigation.

CHAPTER 6: DISCUSSION

In previous chapters, we present the architecture and experiment results on our proposed HiBi model. As a simple, unsupervised statistical model, HiBi has the benefit of being interpretable and extensible. However, the experiments on English corpora has also revealed several limitations of HiBi. In this chapter, we will discuss a few problems with the current architecture and suggest several directions for future work.

6.1 LIMITATION OF THE MODEL

First of all, the entire HiBi is formulated under the bigram assumption, which is the smallest possible unit to perform associative learning. While the hierarchical structure allows the model to expand the effective window beyond 2 characters, it is still hard, if not impossible, to learn dependency across the longer sequence. For example, in the following sentence segmentation:

> “the |fu|lton| |count|y| |grand| |jury| |said| |friday| |an| |investigation| of |at|lant|a|’s| |recent| |prim|ary| |elect|i|on| |produce|d| |“ no| |evidence| |”| |that| |any| |ir|regular|ities| |took| |place| . ”

Using bigrams as the basis of language modeling could be problematic in this case, because spaces are also treated as a valid token. As a result, no matter which word we come from, once the model parses the space, all semantic information from the previous context is lost as the parsing for the rest of the sentence depends on $P(t \mid \text{` '})$ only.

Another thing that limits the current model is how the learned tokens are represented. Currently, all tokens are stored and represented in their surface form in the memory. Even though we were able to mine some lexical relationships between the learned tokens, when the model performs the chunking process, all of them were treated as independent units. This has the same problem as using one-hot vectors in neural networks, and people have been using a dense vector as word embeddings to mitigate the problem [36, 37, 38].

6.2 FUTURE DIRECTIONS

To overcome the problem of bigram assumption that we have in this model, which greatly limits the amount of context it is able to capture, one of the possibilities is to allow for a larger context window (e.g., by using trigrams) when doing segmentation. Indeed, since humans' working memory could retain around 3 chunks of items [39], a slightly larger context would be a reasonable step to go [1].

As for the latter problem about the representation of tokens, one approach to replacing the surface structure is to introduce clustering, or classification of individual tokens, whenever a new concept is formed. This is essentially equivalent to introducing the hidden states as in the HMM models, with the difference that the number of clusters should not be predefined in advance and should be able to grow and shrink from time to time. The transition probability should be computed from cluster to cluster, with each cluster being a semantically related set of tokens (e.g., synonyms). It would also support the spreading activation theory in cognitive science, where semantically related ideas are stored interconnected in a network [40].

It would also be interesting to compare HiBi with human subjects to see whether our proposed model indeed matches how humans process information. To avoid introducing biases, the experiment should be done with symbols unknown to human subjects. The same sequence of symbols should be given to both groups, and we could have the human subject group the symbols which they believe should go together.

Finally, the learned token representation could potentially be used in other downstream tasks as a substitute for the word-level representation. For example, we could apply the HiBi to segment each document into a sequence of tokens, perform text categorization at the token level, and compare its performance with the traditional bag-of-words approach.

[1]Miller originally reported the working memory limit to be 7 ± 2 chunks, but this number was brought down by further research.

CHAPTER 7: CONCLUSION

We introduce the HiBi model, which is a hierarchical bigram model design for learning meaningful representations from unlabeled sequential data. We explain the motivation of designing HiBi in Chapter 1 and describe the basis of our model, namely, the bigram language model, in Chapter 2. A few related works are discussed in Chapter 3 as reference points for our work. HiBi's architecture is discussed in detail in Chapter 4, where we first illustrate the overall idea at a high level, then visit each of the components and explain our choice of design. In Chapter 5, we run our HiBi with a few real-world datasets to evaluate its ability to extract meaningful tokens from raw corpora. We then discuss a few limitations of HiBi and several possible ways to improve it as future directions in Chapter 6.

The HiBi model was originally inspired by Pavlov's classical conditioning experiment, in which dogs learn to pair conditioned stimulus with an unconditioned response after several trials [41]. Even though the experiment was actually about a slightly different aspect of learning, Pavlov's experiment results, together with Hebb's theory on synaptic plasticity [26], still led us wondering whether humans learn to associate two tokens t_a and t_b if t_b always happen after t_a.

Another theory that further affirms our postulation is the chunking theory for human information processing [39]. It states that humans have limited working memory, which can only retain around 2 or 3 bits of information. The reason we seem to be able to remember much more is that our information processing system keeps breaking down inputs into familiar "chunks," so that each of them requires minimum effort to retain. A related theory is about the use of pattern recognition as expertise, where, for example, chess players can learn to recognize complex configurations of chess pieces as a single chunk and retrieve relevant strategies based on it through years of training. Therefore, we believe that a human-like cognitive model should have a similar capacity to perform chunking on the inputs based on experience, which led us to build the HiBi model.

As a model inspired by human information processing theories, HiBi has shown to be able to extract meaningful information from raw texts with no supervision. We sincerely hope that our work could inspire more people to consider incorporating cognitive theories when designing new architectures.

It is great to have a complex model that does the work, but isn't it even better if there is a simpler one that is as effective?

REFERENCES

[1] Apollodorus and J. G. Frazer, *Apollodorus: The library.* Harvard University Press, 1995.

[2] Merriam-Webster Online, "Merriam-Webster Online Dictionary," 2009. [Online]. Available: http://www.merriam-webster.com

[3] J. H. Moor, "The dartmouth college artificial intelligence conference: The next fifty years," *AI Magazine*, vol. 27, pp. 87–91, 2006.

[4] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[5] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," 2019.

[6] D. Deming, "The robots are coming. prepare for trouble." *The New York Times*, Jan 2020, accessed: 2020-04-29. [Online]. Available: https://www.nytimes.com/2020/01/30/business/artificial-intelligence-robots-retail.html

[7] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, "Albert: A lite bert for self-supervised learning of language representations," in *International Conference on Learning Representations*, 2020. [Online]. Available: https://openreview.net/forum?id=H1eA7AEtvS

[8] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, "Squad: 100,000+ questions for machine comprehension of text," *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, 2016. [Online]. Available: http://dx.doi.org/10.18653/v1/D16-1264

[9] H. A. Simon, *Models of thought.* New Haven: Yale University Press, 1979.

[10] D. Jurafsky and J. H. Martin, *Speech and language processing: an introduction to natural language processing, computational linguistics, and speech recognition*, 2nd ed., ser. Prentice Hall series in artificial intelligence. Upper Saddle River, N.J: Pearson Prentice Hall, 2009, oCLC: 213375806.

[11] S. Katz, "Estimation of probabilities from sparse data for the language model component of a speech recognizer," *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 35, pp. 400 – 401, 04 1987.

[12] D. G. Hays, "Dependency Theory: A Formalism and Some Observations," *Language*, vol. 40, no. 4, p. 511, Oct. 1964. [Online]. Available: https://www.jstor.org/stable/411934?origin=crossref

[13] E. Charniak, *Statistical Language Learning*, 1st ed. Cambridge, MA, USA: MIT Press, 1996.

[14] A. J. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Transactions on Information Theory*, vol. 13, no. 2, pp. 260–269, 1967.

[15] S. Fine, Y. Singer, and N. Tishby, "The hierarchical hidden markov model: Analysis and applications," *Mach. Learn.*, vol. 32, no. 1, p. 4162, July 1998. [Online]. Available: https://doi.org/10.1023/A:1007469218079

[16] E. A. Feigenbaum and H. A. Simon, "A theory of the serial position effect," *British Journal of Psychology*, vol. 53, no. 3, pp. 307–320, Aug. 1962. [Online]. Available: http://doi.wiley.com/10.1111/j.2044-8295.1962.tb00836.x

[17] H. A. Simon and E. A. Feigenbaum, "An information-processing theory of some effects of similarity, familiarization, and meaningfulness in verbal learning," *Journal of Verbal Learning and Verbal Behavior*, vol. 3, no. 5, pp. 385–396, Oct. 1964. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0022537164800074

[18] L. W. Gregg and H. A. Simon, "An information-processing explanation of one-trial and incremental learning," *Journal of Verbal Learning and Verbal Behavior*, vol. 6, no. 5, pp. 780–787, Oct. 1967. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0022537167800860

[19] S. McMains and S. Kastner, "Interactions of Top-Down and Bottom-Up Mechanisms in Human Visual Cortex," *Journal of Neuroscience*, vol. 31, no. 2, pp. 587–597, Jan. 2011. [Online]. Available: http://www.jneurosci.org/cgi/doi/10.1523/JNEUROSCI.3766-10.2011

[20] K. W. Church and P. Hanks, "Word association norms, mutual information, and lexicography," *Computational Linguistics*, vol. 16, no. 1, pp. 22–29, 1990. [Online]. Available: https://www.aclweb.org/anthology/J90-1003

[21] P. Brass, *Advanced data structures.* Cambridge ; New York: Cambrige University Press, 2008, oCLC: ocn221147457.

[22] A. Tversky and D. Kahneman, "Availability: A heuristic for judging frequency and probability," *Cognitive Psychology*, vol. 5, no. 2, pp. 207–232, Sep. 1973. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/0010028573900339

[23] M. A. Just and P. Carpenter, "A theory of reading: from eye fixations to comprehension." *Psychological review*, vol. 87 4, pp. 329–54, 1980.

[24] T. Bever, *The Cognitive Basis for Linguistic Structures*, 01 1970, pp. 279–352.

[25] M. Medress, F. Cooper, J. Forgie, C. Green, D. Klatt, M. O'Malley, E. Neuburg, A. Newell, D. Reddy, B. Ritea, J. Shoup-Hummel, D. Walker, and W. Woods, "Speech understanding systems: Report of a steering committee," *Artificial Intelligence*, vol. 9, no. 3, pp. 307 – 316, 1977.

[26] D. O. Hebb, *The organization of behavior: a neuropsychological theory.* Mahwah, N.J: L. Erlbaum Associates, 2002.

[27] S. Lowel and W. Singer, "Selection of intrinsic horizontal connections in the visual cortex by correlated neuronal activity," *Science*, vol. 255, no. 5041, pp. 209–212, 1992. [Online]. Available: https://science.sciencemag.org/content/255/5041/209

[28] G. Bouma, "Normalized (pointwise) mutual information in collocation extraction," *Proceedings of the Biennial GSCL Conference 2009*, 01 2009.

[29] A. El-Kishky, Y. Song, C. Wang, C. R. Voss, and J. Han, "Scalable topical phrase mining from text corpora," *Proceedings of the VLDB Endowment*, vol. 8, no. 3, p. 305316, Nov 2014. [Online]. Available: http://dx.doi.org/10.14778/2735508.2735519

[30] T. Johnson and D. Shasha, "2q: A low overhead high performance buffer management replacement algorithm," in *Proceedings of the 20th International Conference on Very Large Data Bases*, ser. VLDB 94. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1994, p. 439450.

[31] H. Ebbinghaus, *Memory: A contribution to experimental psychology.* New York: Teachers College Press, 1913. [Online]. Available: http://content.apa.org/books/10011-000

[32] E. Loper and S. Bird, "Nltk: The natural language toolkit," in *Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics - Volume 1*, ser. ETMTNLP 02. USA: Association for Computational Linguistics, 2002. [Online]. Available: https://doi.org/10.3115/1118108.1118117 p. 6370.

[33] "Project Gutenberg." [Online]. Available: http://www.gutenberg.org/

[34] W. N. Francis and H. Kucera, "Brown corpus manual," Department of Linguistics, Brown University, Providence, Rhode Island, US, Tech. Rep., 1979. [Online]. Available: http://icame.uib.no/brown/bcm.html

[35] S. Jiang and C. Zhai, "Random walks on adjacency graphs for mining lexical relations from big text data," *Proceedings - 2014 IEEE International Conference on Big Data, IEEE Big Data 2014*, pp. 549–554, 01 2015.

[36] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," 2013.

[37] J. Pennington, R. Socher, and C. Manning, “Glove: Global vectors for word representation,” in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP).* Doha, Qatar: Association for Computational Linguistics, Oct. 2014. [Online]. Available: https://www.aclweb.org/anthology/D14-1162 pp. 1532–1543.

[38] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov, “Bag of tricks for efficient text classification,” in *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers.* Association for Computational Linguistics, April 2017, pp. 427–431.

[39] G. A. Miller, “The magical number seven, plus or minus two: some limits on our capacity for processing information.” *Psychological Review*, vol. 63, no. 2, pp. 81–97, 1956. [Online]. Available: http://doi.apa.org/getdoi.cfm?doi=10.1037/h0043158

[40] J. R. Anderson, “A spreading activation theory of memory,” *Journal of Verbal Learning and Verbal Behavior*, vol. 22, no. 3, pp. 261 – 295, 1983. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0022537183902013

[41] P. I. Pavlov (1927), “Conditioned reflexes: An investigation of the physiological activity of the cerebral cortex,” *Annals of Neurosciences*, vol. 17, no. 3, pp. 136–141, July 2010. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4116985/