Natural Language Processing

Assignment 1

Fatemeh Nadi 810101285

March 16, 2023

Problem 1

Please see this file: "CA1_NLP_Q1.ipynb"
Code and explanation are provided

A.
Cleaning and other pre-processing

- Converting your text to lower case. Farsi does not require this step.
- Character replacement. Change Arabic to Farsi.
- Punctuation and non-alphanumeric character removal.
- It is not necessary to include English paragraphs and numbers in this text. remove these.
- Prepare text for next steps by applying some rules.

Let's take a look at the statistics of the corpus:

	Number of sentences	Number of words	Number of unique words	Average word length	Average word per sentence
harry potter	5697	60451	8963	4.536253	16.203616

B.
There are some words that occur together more often than others. Examples include Sky High, Do or Die, Best Performance, Heavy Rain, etc. Therefore, sentiment analysis may require identifying a pair of words within a text document. To maintain their current sequences, we must first generate word pairs from the existing sentence. Such pairs are called bigrams. The NLTK library in Python has a bigram function that helps us generate these pairs.

C.
The following may happen if the Laplace smoothing method is not used:

- Predictions that are very wrong because of incorrect information or RARE values that exist in the data.
- Predictions may be inaccurate if new occurrences are unknown in the data.
- Smoothing can improve model accuracy by removing the correlation between features.

D.
Sentences produced using bigram.

۱. هیچ وقت هایی که هرجا می‌رود در همه‌ی مهاجمین از هوای ما همه پول نمی‌دم
۲. رفت به سوراخ شد با شنل سبزی به سوراخ شد با شنل سبزی به سوراخ شد با شنل سبزی به سوراخ شد
۳. آن‌قدر ادامه بدین! » . آن‌قدر ادامه بدین! » . آن‌قدر ادامه بدین
۴. لبخند شیطانی رو شکر کن بسته‌اند در مجاور دیوار می‌کند بیش از غم عمیقی نهفته_بود گفت: -
۵. و وحشی و وحشی و وحشی و وحشی و وحشی و وحشی و وحشی و وحشی و

E.
Sentences produced using trigram

۱. انداخت . بیش از آن به بعد توی اتاق دومی دادلی باشی . اونا
۲. شنل می‌توانست به سرعت جلو رفتند و سالن عمومی شدند . هاگرید لبخندی بر لب هاگرید نشست و منتظر می‌مونم تا فیلچ بیاد و
۳. گفت و همراه نویل به چهره‌های خطاکار آن‌ها نگاه کرد که کنار پاشه چیه ؟ پیش از مسابقه چشم بردارد به نویل گفت :
۴. . آن خانه یک اتاق‌خواب بیش‌تر نداشت . آقای اولیوندر ادامه داد: - اسلیترین صاحب توپه.
۵. سوی رون رفت که به طرف صدای بدعنق می‌رفت می‌شنیدند . اسنیپ روبه روی شبحی که یقه‌ی لباس

Sentences produced using five-gram.

۱. به خودش اشاره کرد . رون گفت: - تو به ده تا مالفوی می‌ارزی . مگه کلاه گروهبندی
۲. تکیه داد و مشتاقانه به هری خیره شد . پس از نیم ساعت
۳. مثل مار درازی در هوا پیچ و تاب می‌خورد به سرعت از آن جا رفت . هرمیون که در فکر بود گفت: -
۴. شنیدن حرهای هاگرید دوربین او را گرفت و شروع به پاک کردن بینیاش کرد . پسر
۵. از این برخورد ناگهانی ناراحت شده_باشد . پیرمرد لبخندی به پهنای صورتش زد و با صدای زیری که باعث جلب توجه رهگذرها

F.
It is the number of words that are considered for predicting the next word in a sequence that makes the difference between sentences generated using bigram models, trigram models, and five-gram models.

- A bigram model considers pairs of adjacent words, i.e., the conditional probability of a word given its preceding word.
- The trigram model considers triplets of adjacent words, i.e., the conditional probability of a word given its two preceding words.
- Five-gram models take five adjacent words into account.

With increasing N-gram order, the model captures more context and dependencies between words, resulting in more accurate and coherent sentences. In addition to data sparsity and overfitting, higher-order models require more time and computing resources.

In the first model, both consecutive words seem logically combined. Likewise, meaningful sentences are produced in five grams rather than two and three grams as the number of grams increases.

G.
We have some test data, m sentences $s_1, s_2, ..., s_m$
We could look at the probability under our model $\prod_{i=1}^{m} p\left(s_i\right)$

Or the log probability:

$$log \prod_{i=1}^{m} p\left(s_i\right) = \sum_{i=1}^{m} log\, p\left(s_i\right)$$

$$perplexity \;=\; 2^{-l};\; where\; l = \frac{1}{M} \sum_{i=1}^{m} log\, p\left(s_i\right)$$

Due to our model training with Harry Potter's book, our model generates the first sentence with a higher probability, so the perplexity of the first sentence is lower.

Problem 2

Please see this file: "CA1_NLP_Q2.ipynb"
Code and explanation are provided

Tokenizers break unstructured data and natural language text into discrete pieces of information. Using the token occurrences in a document as vectors can represent the document directly.
The tokenization process can be used to separate sentences, words, characters, or subwords. The process of splitting text into sentences is called sentence tokenization. Word tokenization refers to the process of tokenizing words.

A.
White Space Tokenization:
Using whitespace within a string as a "delimiter" between words is the simplest way to tokenize text. In order to accomplish this, Python's split function can be used or the 'nltk.tokenize' library can be used.

Spacy Tokenizer:
There are several steps involved in tokenizing a text in Spacy. Tokenizers are designed to work in various languages and text types and combine rule-based and statistical approaches to achieve high accuracy and performance.

- In the first step, the tokenizer separates the text into whitespace, similar to the split() function, and converts any non-standard characters or symbols to their standard Unicode equivalents.

- Tokenizers then check whether the substring matches the exception rules. At this stage, we use rule-based and statistical methods that consider punctuation, capitalization, and other contextual features. Each language and text type has rules and patterns for common cases like contractions, abbreviations, compound words, and special characters.

- The tokenizer normalizes and standardizes the tokens once they are identified. All tokens are converted to lowercase, stop words are removed, and stemming or lemmatization is used to reduce inflectional forms to their base forms. Post-processing ensures that tokens are clean, standardized, and ready to be analyzed.

(BPE)Subword Tokenization:
As a sub-word and morphological tokenizer, BPE merges adjacent byte pairs based on their frequency in a training corpus. BPE takes a pair of tokens (bytes), looks at their frequencies, and merges the pair with the highest frequency. Each step of the process is greedy for the highest combined frequency.
To start, we should detect all words in the corpus and add a symbol to indicate the end of each word " $< /w >$ ".
Next, we make a vocabulary of all bytes (symbols and characters) in these words; finally, we count the frequency of consecutive byte pairs, and merge the most frequent one. Until a token limit is reached, we should repeat this procedure.

For the given corpus the steps of BPE is like this:

low low low low low lower lower widest widest widest newest newest newest newest newest.

- The " $< /w >$ " symbol is concatenated to all words.

- All words will be split into symbols(characters):
  l o w $< /w >$ *l o w* $< /w >$ *l o w* $< /w >$ *l o w* $< /w >$ *l o w* $< /w >$ *l o w e r* $< /w >$ *l o w e r* $< /w >$ *w i d e s t* $< /w >$ *w i d e s t* $< /w >$ *w i d e s t* $< /w >$ *n e w e s t* $< /w >$ *n e w e s t* $< /w >$ *n e w e s t* $< /w >$ *n e w e s t* $< /w >$ *n e w e s t* $< /w >$ *n e w e s t* $< /w >$ *n e w e s t* $< /w >$

- The vocabulary is created by unique symbols of all words in the corpus:
  Vocabulary =$\{l, o, w, e, r, i, d, s, t, n, < /w >\}$

- Now we compute frequency of every consecutive pair and merge the most frequent one.

  "l o w $< /w >$" : 5
  "l o w er $< /w >$" : 2
  "w i d e s t $< /w >$" : 3
  "n e w e s t $< /w >$" : 5

  With a frequency of 8, "es" is the most frequent pair. Finally, there is "est" with a frequency of 8, and so on.
  Vocabulary = $\{l, o, w, e, r, i, d, s, t, n, < /w >, es, est, est < /w >,$
  $lo, low, ne, new, newest < /w >, low < /w >$

B.

algorithm	# Tokens	
	English	Persian
White Space	78443	96294
Spacy	102406	125677
BPE	100012	100932

C.
Input : *This question is about tokenization and shows several tokenizer algorithms.Hopefully, you will be able to understand how they are trained and generate tokens.*

White Space : This, question, is, about, tokenization, and, shows, several, tokenizer, algorithms.Hopefully,, you, will, be, able, to, understand, how, they, are, trained, and, generate, tokens. (23)

Spacy : This, question, is, about, tokenization, and, shows, several, tokenizer, algorithms, ., Hopefully, ,, you, , will, be, able, to, understand, how, they, are, trained, and, generate, tokens, . (28)

BPE : This, question, is, about, to, ken, i, z, ation, and, shows, several, to, ken, i, z, er, al, gor, ith, ms, ., Hopefully, ,, you, will, be, able, to, understand, how, they, are, trained, and, g, en, er, ate, to, kens, . (42)

Input : "این سوال در مورد قطعه بندی جملات است و چندین الگوریتم توکنایز کردن متن را نشان می دهد. امیدواریم بتوانید نحوه آموزش آنها و تولید توکن ها را درک کنید."

White Space (۳۰)

این, سوال, در, مورد, قطعه, بندی, جملات, است, و, چندین, الگوریتم, توکنایز, کردن, متن, را, نشان, می, دهد., امیدواریم, بتوانید, نحوه, آموزش, آنها, و, تولید, توکن, ها, را, درک, کنید.

Spacy (۳۳)

این, سوال, در, مورد, قطعه, بندی, جملات, است, و, چندین, الگوریتم, توکنایز, کردن, متن, را, نشان, می, دهد, ., امیدوار, یم, بتوانید, نحوه, آموزش, آنها, و, تولید, توکن, ها, را, درک, کنید, .

BPE (۶۳)

ا, <unk>, بند, عه, قط, مورد, در, ال, سو, ن, <unk>, چند, و, است, لات, جم, <unk>, گور, ال, ن, <unk>, ا, کن, تو, تم, د, ن, حو, ه, آموزش, <unk>, م, ب, توان, <unk>, دو, ار, <unk>, ام, A, دهد, <unk>, ز, کردن, متن, را, نشان, م, <unk>, آنها, و, تول, <unk>, کن, درک, را, ها, کن, تو, د, <unk>, د, A

Conclusion

When punctuation like ”.” or ”,” is attached to the previous word in the White Space method, it cannot detect it. Furthermore, we sometimes need sub-words in languages like Persian, so we miss some information about the text.

In the English sentence, the white space method does well, but just in ”algorithms.Hopefully,” it cannot detect true tokens. This time, the Spacy method works perfectly and correctly detects all tokens. The BPE method is not suitable because the train data or corpus is related to Harry Porter’s book, but this text is unrelated, so the train and test don’t belong to the same genre and couldn’t work well together.

As before, the white space method cannot identify valid tokens in some Persian words. In Persian text, some words are separated, but still a word, like some verbs, and this method cannot detect this. In this case, Spacy works better than others and correctly detects some tokens, but still cannot detect all of them. As the train data or corpus is related to Harry Porter’s book, the BPE method is not suitable. Since train and test are unrelated, they couldn’t work well together since they don’t belong to the same genre. This model didn’t train for this test text, so we have so many unknown words. The text

(train and test) is not clean. For example, we must change Arabic characters to Persian; now, for all ”ye characters” in the text, this model returns unknown.