A. **Specific problem definition** :

In this homework, the task is to search for corpuses in nltk or online corpuses to collect sentences that contain two or more heteronyms (heteronym is one of two or more homographs that differ in pronunciation and meaning). The next task is to rank the sentences that we have found basing on 3 criteria: (1) sentence with more occurrences of homograph has higher ranking, (2) sentence with homograph is ranked higher than a sentence without homograph, (3) sentence containing heteronyms with the same part-of-speech has higher ranking than those with different part-of-speech. And priority of these criteria are (1)>(2)>(3). And after that, we need to annotate pronunciation of heteronyms in each sentence.

B. Discuss the result:

a. Idea for the code to prove the reasonability of my result:

I. There are 2 main issues that we have to deal with: search for possible words that can be heteronyms and finding way to annotate pronunciation for each heteronym inside a sentence. To deal with the first issue, after spending time for experimenting, I realize that most of words that have at least two meanings and at least two way of pronunciation are more likely to be heteronyms, therefore, to find for heteronyms, I will find for the words that have at least two meanings and at least two pronunciations. Firstly, i initiate an empty list named list_heter, then I search for words in nltk.cmudict to find words that have at least two pronunciations and add them to the list collect_heter. There is one thing to notice, that cmudict is a little bit outdated, therefore there are some words with just one pronunciation, but appearing to have 2 or more pronunciations in cmudict. Thus, I have to write a filter function to avoid adding them to the list. Next, to have a more accurate collection, I use another dictionary named Wiktionary, and in the implementation, I write a parser that can parse word content from Wiktionary to JSON format, and with this parser, we can extract pronunciations of a word from the Wiktionary.

Then, with each words from list list_heter , I will extract pronunciations of these words from Wiktionary and check if their number of pronunciations are at least two or not, if not then I would exclude them from list collect_heter. Now, we have a list of words that have at least 2 ways of pronunciation, and we can trust this list because we have filtered it through two dictionaries: cmudict and Wiktionary, and furthermore, Wiktionary is an online updated dictionary, so contents extracting from it are reliable. Next, to deal with meaning issue, we can use nltk.corpus.wordnet because with any word A, we can have B= nltk.corpus.wordnet.synsets(A), then B is a list of synsets, and the number of synsets in B is the number of meanings of word A. Therefore, with each word in list collect_heter, I will use wordnet to check if its number of meaning is at least two, and if its number of meanings is less than two, I would exclude this word from list list_heter. From now on, we have list list_heter containings words with at least two meanings and two pronunciations. I will consider this list as a list containing heteronyms for the next parts.

II. With second task, searching for sentences containing two or more heteronyms, I initiate a list collect_sentence and then, I use 4 corpuses Brown, Webtext, Inaugural, Reuters of nltk to collect sentences with at least two words appearing in the list list_heter (defined in part I) and now we have the list collect_sentence containing sentences with at least two heteronyms.

III. The next task is to give ranking to these sentences. To do this, I use the following formula to give score for each sentence:

Score = 100*score_1_2 + score_3 **(a)**

where score_1_2 is the score for priority 1 and 2, and it is the number of occurrences of homograph inside a sentence, then obviously, sentences containing more occurrences of homograph would have higher score than ones with fewer, and sentences with occurrences of homograph would have higher score than sentences not containing homograph. Score_3 is the score for priority 3, and it is the number of heteronyms with same part-of-speech information inside the sentence, then obviously, sentences containing heteronyms with same part-of-speech would have higher score. To keep the priority (1)>(2)>(3), I give the weight 100 for score_1_2 and weight 1 for score_3 in formula (a).

IV. The final task is to annotate pronunciation for each heteronym inside the sentence. To do this, I use Wiktionary because it has pronunciation information for vocabularies. The information page of Wiktionary for one vocabulary A is organized as following: one vocabulary can have many Etymologies, so the content would be divided according to etymologies of A and content is divided into the form Ety_1,Ety_2,…Ety_n where Ety_k correspond to kth etymology of the vocabulary A. Each of Ety_1,Ety_2,… has a unique pronunciation for A, and pronunciations associated with Ety_1,Ety_2,… are different from each other.

Furthermore, each of Ety_1,Ety_2,… contains a list of different definitions for A. Assuming these list of definitions are Def_1,Def_2,…Def_k , then they are list of definitions of A and Def1 is contained in Ety_1 part,Def_2 is contained in Ety_2 part,… And basing on this structure of Wiktionary, my idea for annotating pronunciation is : Assume we have sentence X, we use nltk.wsd.lesk to get definition string of each heteronym inside X, assuming heteronym H has definition string K, then I write a function to measure the semantic similarity of two strings and use this function to measure the semantic similarity of definition string K and all definition strings contained in definition lists Def_1,Def_2,…Def_k. Assume L is a string contained in a list Def_n that gives highest score when measuring semantic similarity with string K, then it is highly true that the definition string of heteronym H is contained in Def_n, and we already know that Def_n is contained in part Ety_n so I use the pronunciation associated with Ety_n to annotate for heteronym H. In this way, we can annotate pronunciation for all heteronyms of a sentences.

b. **Evaluating result**: Looking at 30 sentences with highest ranking, I observe that the result seems to be quite reasonable because most of them contains heteronyms and there are about 10 distinct heteronyms appearing in these 30 sentences. In term of ranking, it is unfortunate that sentences containing at least 2 occurrences of homographs seem rare and even I have use 4 corpuses Brown, Reuter, Webtext, Inaugural of nltk, the result still does not contain any sentences with this condition, therefore, the score for priority 1 is not applied at all. However, the result of my code implementation can find sentences with one occurrence of homograph, and they are ranked very high (in fact, all of 30 sentences with highest ranking, each of them has one occurrence of homograph). This is a very reasonable ranking because it agrees with the priority 2. Furthermore, score of priority 3 also affect to the ranking of my result, among 30 sentences with highest ranking, some of them get score for this priority because containing heteronyms with same part-of-speech. In term of pronunciation, I observe many sentences have correct pronunciation annotation for their heteronyms. However, the result still need more improvement, for example, some sentences do not contain any heteronyms, and furthermore, my result do not have any sentence that contain at least 2 occurrences of homograph and last but not least, the pronunciation annotation is not good for some sentences.

c. **Improvement**: My implementation creates a list named list_heter to collect heteronyms by using cmudict and Wiktionary, however, the result for this list still contains some words that are not heteronyms. It maybe because I use the filter condition: a word has at least two meanings and two pronunciations would be considered as heteronyms and would be added to the list. By experimenting, I realize that this condition is quite good and it can help me to find out many heteronyms and it is also quite strict because it helps to exclude most of words that are not heteronyms. However, I think that if I can improve this filter condition to make it more strict, I can exclude more words that are not heteronyms from list_heter. Furthermore, my implementation use the dictionary nltk.cmudict, but this dictionary seems outdated, thus, if I can find a more updated dictionary, I believe that the result can be improved. Secondly, when I collect sentences with at least 2 homographs, I use 4 corpuses Brown, Webtext, Inaugural, Reuters and cannot use more because it would make the running time become large, thus, if I have a better laptop with faster speed, I can use more corpuses, not only nltk corpuses but also online corpuses , and in this way, I believe that the result would have better sentences, for example ones that contain at least 2 occurrences of homograph. Lastly, to annotate pronunciations for each heteronyms inside a sentence, I write a function to measure the similarity of 2 sentences as mentioned above, however this function is just based on the idea of turning 2 strings into vectors and measure cosine angle of these two vectors, but from what I observe, this function is not really effective and need more improvement. Because I already use Wiktionary dictionary in my implementation, I do not want to use one more external model because 2 external models seem too much, but for the purpose of improving the result, using external models that have been trained to measure semantic similarity of two sentences would help my implementation to give better pronunciation annotation for heteronyms inside a sentence.

C. **External environment for implementation**: parser for Wiktionary dictionary

In file .py, to run the code, first, we need to write install this external model by writing this line in terminal command :

```
pip install wiktionaryparser
```