19-06-2023

**Chapter 1: Language Processing and Python**

First 2 sections of the 1st chapter are primarily concerned with setting up of the python and nltk environment. I had my python environment set and updates already. Following the instructions of the book and a little help from stack overflow, I completed setting up of the nltk environment as well.

**Searching Text:**

A concordance permits us to see words in context.

Syntax: text\_name. concordance(“the\_word\_that\_we\_need\_to\_find\_similarity\_for”)

\*\* I have practiced practically to find the concordance of few text words. I have attached the output screenshots as well \*\*

Similar () and common\_contexts () are other functions whose working is quite similar.

**Dispersion Plots:**

It is one thing to automatically detect that a particular word occurs in a text, and to display some words that appear in the same context. However, we can also determine the location of a word in the text: how many words from the beginning it appears. This positional information can be displayed using a dispersion plot. Each stripe represents an instance of a word, and each row represents the entire text.

**Counting Vocabulary:**

Using of len () function helps us find out the length of a text from start to finish, in terms of words and punctuation marks that occur.

A **token** is the technical name for a sequence of characters—such as hairy, his, or :)—that we want to treat as a group. When we count the number of tokens in a text, say, the phrase to be or not to be, we are counting occurrences of these sequences.

Here, the key point to note is that, len (text3) returns the full length of the text file, i.e., including multiple repetitions of occurrences of words/tokens.

So, to have a non-repeated length of all the tokens used in the text, we use **len (set (text3)).**

By wrapping sorted () around the Python expression set(text3) , we obtain a sorted list of vocabulary items, beginning with various punctuation symbols and continuing with words starting with A. All capitalized words precede lowercase words. We discover the size of the vocabulary indirectly, by asking for the number of items in the set, and again we can use len to obtain this number. Although it has 44,764 tokens, this book has only 2,789 distinct words, or “word types.” A word type is the form or spelling of the word independently of its specific occurrences in a text—that is, the word considered as a unique item of vocabulary. Our count of 2,789 items will include punctuation symbols, so we will generally call these unique items types instead of word types.

**Lexical Richness:**

It is the measure of how many times a word on average occurs in a text.

**Counting specific words in a text:**

This can be done using the count () function. This function takes a word as input and returns the number of occurrences of the particular word as the desired output.

**Lists:**

Lists are used to store multiple items in a single variable. Lists are one of 4 built-in data types in Python used to store collections of data, the other 3 are Tuple, Set, and Dictionary, all with different qualities and usage.

* We can ask for its length.
* We can even apply our own lexical\_diversity () function to it.

1. Concatenation:

Adding two lists creates a new list with everything from the first list, followed by everything from the second list. The special use of the addition operation is called concatenation; it combines the lists together into a single list. We can concatenate sentences to build up a text.

1. Appending:

Process of not adding two lists but adding an element to an existing list. As a result, the list itself is updated.

1. Indexing Lists:

The number that represents the position of an element in the list, is the item’s index. Indexing in python starts from 0 (first element) till n-1 (last element), where n is the total number of elements existing in the list.

1. Slicing of Lists:

Python permits us to access sub lists as well, extracting manageable pieces of language from large texts, a technique known as slicing.

**Frequency Distributions in Text:**

Tells us about the frequency of each vocabulary item in the text. It is a “distribution” since it tells us how the total number of word tokens in the text are distributed across the vocabulary items. Since we often need frequency distributions in language processing, NLTK provides built-in support for them.

**Fine-Grained Selection of Words:**

This includes conditions which the developer is looking for in the text. This is the process of selecting tokens out a text which satisfies a particular condition. For example, returning a set of tokens whose word length is greater than 15 characters, and so on.

It is a modest but important milestone: a tiny piece of code, processing tens of thousands of words, produces some informative output.

**Collocations and Bigrams:**

A collocation is a sequence of words that occur together unusually often. Thus, red wine is a collocation, whereas the wine is not. A characteristic of collocations is that they are resistant to substitution with words that have similar senses; for example, maroon wine sounds very odd.

To get a handle on collocations, we start off by extracting from a text a list of word pairs, also known as **bigrams**. This is easily accomplished with the function bigrams (): >>> bigrams (['more', 'is', 'said', 'than', 'done']) [('more', 'is'), ('is', 'said'), ('said', 'than'), ('than', 'done')].

**Collocations are essentially just frequent bigrams, except that we want to pay more attention to the cases that involve rare words.**

**Automatic Natural Language Understanding:**

Natural language processing (NLP) offers the potential to tackle the challenge of building intelligent machines. While exploring language from a bottom-up approach using texts and Python programming language, the focus now shifts towards building useful language technologies. At a practical level, search engines have been vital for navigating the vast amount of information on the web, but they have limitations in providing specific answers to complex questions. Automating this process requires various language processing tasks such as information extraction, inference, and summarization, which currently exceed our capabilities in terms of scale and robustness.

On a philosophical level, language understanding is a crucial aspect of intelligent behavior, and it has long been considered a difficult challenge in artificial intelligence. However, advancements in NLP technologies and the availability of robust methods for analysing unrestricted text are making this goal more attainable.

1. **Word Sense Disambiguation:**

Word sense disambiguation aims to determine the intended meaning of a word in a given context. For example, the words "serve" and "dish" have multiple senses. By analysing the context in which they are used, such as in the sentence "he served the dish," we can infer that both "serve" and "dish" refer to their food-related meanings rather than other possible interpretations. This context-based disambiguation relies on the observation that nearby words often have closely related meanings. Similarly, the word "by" can have different meanings depending on the context, such as indicating an agentive relationship ("the book by Chesterton"), a locative relationship ("the cup by the stove"), or a temporal relationship ("submit by Friday"). Understanding the meaning of the italicized word in each example helps us interpret the overall meaning of the sentence.

1. **Pronoun Resolution:**

A deeper level of language understanding involves determining the subjects and objects of verbs, or in other words, figuring out "who did what to whom." While this may seem straightforward, it can be challenging. For example, in the sentence "the thieves stole the paintings," it is easy to identify who performed the stealing action. However, when considering subsequent sentences like "They were subsequently sold," "They were subsequently caught," and "They were subsequently found," it becomes more difficult to determine what was sold, caught, or found (as one case is ambiguous).

Addressing this question requires identifying the antecedent of the pronoun "they," which could refer to either the thieves or the paintings. Computational techniques used to tackle this problem include anaphora resolution, which involves determining what a pronoun or noun phrase refers to, and semantic role labelling, which involves identifying how a noun phrase relates to the verb (as an agent, patient, instrument, etc.).

1. **Generating Language Output:**

If we can successfully solve problems related to language understanding, it opens the door to more advanced tasks such as question answering and machine translation. For question answering, a machine should be able to accurately respond to user queries based on a collection of texts. For instance, given the text "The thieves stole the paintings. They were subsequently sold," when asked "Who or what was sold?" the machine should correctly identify that the answer is "The paintings," not "The thieves." This demonstrates the machine's ability to understand the context and accurately infer the intended meaning.

In machine translation, the machine should be able to translate text from one language to another while preserving the meaning. This task requires making correct choices based on understanding the original text. For example, when translating the sentence "The thieves stole the paintings. They were subsequently found" into French, the gender of the pronoun in the second sentence needs to be determined correctly. The translation would be "Les voleurs ont volé les peintures. Ils ont été trouvés plus tard" if the thieves are being found, and "Les voleurs ont volé les peintures. Elles ont été trouvées plus tard" if the paintings are being found. The accurate translation depends on correctly understanding the referent of the pronoun.

In all of these examples, determining word senses, identifying the subject of a verb, and finding the antecedent of a pronoun are crucial steps in establishing the meaning of a sentence. These are the tasks we would expect a language understanding system to be capable of performing.

1. **Machine Translation:**

Machine translation (MT) has long been a sought-after goal in language understanding, aiming to provide high-quality and idiomatic translation between any pair of languages. The origins of MT can be traced back to the early days of the Cold War when the potential for automatic translation gained significant government support, leading to the development of Natural Language Processing (NLP) itself.

While practical translation systems exist today for specific language pairs and some are integrated into web search engines, they still have notable limitations. These shortcomings can be explored using NLTK's "babelizer," which allows submitting a sentence for translation and then translating it back into English. However, after a certain number of iterations or when encountering a previously produced translation, the process stops.

The difficulties in machine translation arise from the ambiguity of words and the need to adjust word order based on the grammatical structure of the target language. Addressing these challenges involves collecting large amounts of parallel texts from sources such as news and government websites, which publish documents in multiple languages. By aligning sentences and identifying corresponding words and phrases, a model can be built for translating new text, leveraging the availability of millions of sentence pairs and potentially bilingual dictionaries.

1. **Spoken Language Systems:**

In the field of artificial intelligence, the Turing Test has been a key measure of intelligence, assessing whether a dialogue system can generate responses indistinguishable from those of a human. However, today's commercial dialogue systems are limited to specific domains, as seen in the example conversation about movie showtimes. These systems can only provide information that has been stored and predefined in their language processing systems, such as movie schedules.

Although it may seem obvious to us, the system needs to understand the user's goals and make appropriate inferences to interact naturally. In the example, the system correctly determines that the user wants to see the movie based on the query about showtimes. Developers of commercial dialogue systems use contextual assumptions and business logic to handle different user requests and provide meaningful responses. By incorporating simple rules, the system can interpret various ways of expressing requests and deliver a useful service.

A typical pipeline for natural language processing (NLP) in dialogue systems consists of language understanding components that convert speech input into a meaningful representation. This is followed by a reverse pipeline that converts concepts into speech output. The system relies on repositories of static information, which serve as data sources for the processing components.

Overall, commercial dialogue systems have limitations in their capabilities but can still perform valuable functions within specific domains by leveraging predefined information and context-specific rules.