Assignment 3

1. **The IOB format categorizes tagged tokens as I, O and B.**

**Why are three tags necessary?** **[1 Mark]**

The IOB (Inside, Outside, Beginning) format is used to annotate sequential data, such as natural language text, with information about the presence and location of named entities or other relevant linguistic features. The IOB format uses three tags:

B (Beginning): Indicates the first token of a named entity or other relevant feature.

I (Inside): Indicates a token that is part of a named entity or other relevant feature, but is not the first token.

O (Outside): Indicates a token that is not part of any named entity or other relevant feature.

**What problem would be caused if we used I and O tags exclusively?** **[1 Mark]**

The use of all three tags is necessary because it allows for unambiguous identification of the boundaries of named entities or other relevant features. For example, consider the sentence "John Smith works at IBM in New York". If we use only I and O tags, the word "IBM" and "New York" would both be labeled with an "I" tag, making it difficult to distinguish between the two entities.

If we used I and O tags exclusively, without the B tag, it would be impossible to determine the beginning of a named entity or other relevant feature, making it difficult to extract and use this information. For example, in the sentence "John Smith works at IBM in New York", without the B tag, it would not be possible to distinguish between the entity "IBM in New York" and the two separate entities "IBM" and "New York". This would make it difficult to perform tasks such as named entity recognition or information extraction accurately.

1. **Write a tag pattern to match noun phrases containing plural head nouns, e.g. "many/JJ researchers/NNS", "two/CD weeks/NNS", "both/DT new/JJ positions/NNS". Try to do this by generalizing the tag pattern that handled singular noun phrases.** **[2 Marks]**

pattern = [{'POS': 'DT', 'OP': '?'}, {'POS': 'JJ', 'OP': '\*'}, {'POS': 'NNS'}]

1. **Pick one of the three chunk types in the CoNLL corpus. Inspect the CoNLL corpus and try to observe any patterns in the POS tag sequences that make up this kind of chunk. Develop a simple chunker using the regular expression chunker nltk.RegexpParser. Discuss any tag sequences that are difficult to chunk reliably**. **[2 Marks]**

By choosing the "NP" (noun phrase) chunk type in the CoNLL corpus and analyze the POS tag sequences that make up this type of chunk.

Looking at the CoNLL corpus, it is observed that noun phrases often begin with a determiner (DT) or possessive pronoun (PRP$), followed by zero or more adjectives (JJ), and end with one or more nouns (NN or NNS) or pronouns (PRP).

simple chunker:

pattern = "NP: {<DT|PRP\$>?<JJ>\*<NN.\*>+}"

chunker = nltk.RegexpParser(pattern)

1. **The Senseval 2 Corpus contains data intended to train word-sense disambiguation classifiers. It contains data for four words: hard, interest, line, and serve. Choose one of these four words, and load the corresponding data:**

|  |  |  |
| --- | --- | --- |
| |  |  | | --- | --- | |  | **>>> from nltk.corpus import senseval**  **>>> instances = senseval.instances('hard.pos')**  **>>> size = int(len(instances) \* 0.1)**  **>>> train\_set, test\_set = instances[size:], instances[:size]** | |

Using this dataset, **build a classifier that predicts the correct sense tag** for a given instance. See the corpus HOWTO at http://nltk.org/howto for information on using the instance objects returned by the Senseval 2 Corpus. **[3 Marks]**

Text

Description automatically generated

Graphical user interface, text

Description automatically generated

1. ☼ Using the movie review document classifier discussed in **Chapter 6.**[**Learning to Classify Text**](https://www.nltk.org/book/ch06.html), generate a list of the 30 features that the classifier finds to be most informative. Can you explain why these particular features are informative? Do you find any of them surprising? **[3 Marks]**

Text

Description automatically generated

A picture containing text

Description automatically generated

1. Consider the following sentence: “SpaceX Starship test flight cancelled minutes before blast-off”. Kindly parse this sentence admitted by the following grammar: **[3 Marks]**

|  |  |  |
| --- | --- | --- |
| |  |  | | --- | --- | |  | **grammar1 = nltk.CFG.fromstring("""**  **S -> NP VP**  **VP -> V NP | V NP PP**  **PP -> P NP**  **V -> "saw" | "ate" | "walked"**  **NP -> "John" | "Mary" | "Bob" | Det N | Det N PP**  **Det -> "a" | "an" | "the" | "my"**  **N -> "man" | "dog" | "cat" | "telescope" | "park"**  **P -> "in" | "on" | "by" | "with"**  **""")** | |
|  |

S -> NP VP

VP -> V NP PP | V NP | V

NP -> "SpaceX" | "Starship" | "test" "flight" | "minutes" "before" "blast-off"

PP -> P NP

V -> "cancelled"

P -> "before"

In this grammar, we have defined a noun phrase for each of the four main words in the sentence: "SpaceX", "Starship", "test flight", and "minutes before blast-off". We have also defined a verb phrase with the verb "cancelled" and optional noun phrase and prepositional phrase complements.

With this grammar, we can parse the sentence as follows:

(S

(NP SpaceX)

(VP

(V cancelled)

(NP (N test) (N flight))

(PP (P before) (NP (N minutes) (P (P before) (NP (N blast-off)))))))