**DESCRIPTION FOR TOOLS AND LIBRARIES USED IN PROJECT**

**1. Tools to extract and process binary formats such as PDF and Docx documents:**

- **PyPDF2:** read pdf file in working directory, get pages from the document, and split text from the document page by page.

- **docx:** read docx file in working directory by paragraphs.

**2. Libraries for Data Structure and Object Manipulation and Display:**

- **pandas:** create and manipulate data frame objects.

- **numpy:** use arrange() function from numpy to rearrange index in data frame.

- **pprint:** prints the formatted representation of object on stream, followed by a newline.

- **random:** create random sample during training process.

**3. Tools for Database Manipulation:**

- **csv:** read csv object and maps the information in each row to a dictionary whose keys are given.

- **sqlite3:** execute SQL query with tools in this library to create table, insert values from csv file with corresponding column, and extract info from table with query.

**4. Natural Language Processing tools from nltk libraries:**

- **nltk:** open source Natural Language Toolkit to perform NLP tasks such as part-of-speech tagging, text classification etc.

- **word\_tokenize:** tokenizer from nltk that is used to divide strings into substrings.

- **pos\_tag:** part-of-speech tagger from nltk to process a sequence of words and attach corresponding pos tag to each word.

- **WordNetLemmatizer from nltk.stem**: WordNetLemmatizer to lemmatize words during text processing.

- **RegexpParser:** uses a set of regular expression patterns to specify the behavior of the parser. This function is used in chunking technique to chunk words in sentence into Noun Phrases, Verb Phrases etc. based on a given chunk grammar which followed by Regular Expression rules.

- **tree2conlltags:** return a list of triplets (word, pos tag, IOB tag), convert a tree to the CoNLL IOB tag format.

- **conlltags2tree:** a reverse of tree2conlltags which will convert the CoNLL IOB format to a tree.

- **tag:** use untag() function from this module to return an untagged version of a tagged sentence.

- **ChunkParserI:** processing interface that will be taken as a base class of self-defined ChunkClassifier class. ChunkClassifier will inherit evaluate() function from this base class.

- **TaggerI:** processing interface that will be taken as a base class of self-defined ChunkTagger class.

- **NaiveBayesClassifier:** Naïve Bayes method is used to classify input feature sets when create sequence classifier Tagger for chunk classification.

- **grammar:** called out FeatureGrammar.fromstring() function from this class to return a feature structure based grammar from the form of a string.

- **parse:** implemented FeatureEarleyChartParser() function() from this class to create a parser suing feature structure based grammar made.

- **Tree from nltk.tree:** Tree() function is used when create self-defined flatten\_childtrees() function to form a new tree structure from a child tree.

- nltk.download('treebank'): download treebank corpus and use it to create training data for sentence segmentation classification.

- nltk.download('maxent\_ne\_chunker'): download missing module.

- nltk.download('words'): download missing module.

- nltk.download('averaged\_perceptron\_tagger'): download missing module.

**5. Self-defined module for Sentence Segmentation**:

- **segment\_sents:** this is a self-defined function to detect the period ‘.’ as a full stop punctuation and segmentize sentences.