

# **BTECH FINAL YEAR PROJECT REPORT**

## **Members:**

<b>Sachin Agarwal</b>	<b>(14/IT/82)</b>
<b>Indraneel Sarkar</b>	<b>(14/IT/21)</b>
<b>Hrishav Mukherjee</b>	<b>(14/IT/78)</b>
<b>Sanjiv Tiwari</b>	<b>(14/IT/69)</b>

# INTRODUCTION

**Natural language processing** (NLP) is a field of computer science, artificial intelligence and computational linguistics concerned with the interactions between computers and human (natural) languages, and, in particular, concerned with programming computers to fruitfully process large natural language corpora. Challenges in natural language processing frequently involve speech recognition, natural language understanding, natural language generation (frequently from formal, machine-readable logical forms), connecting language and machine perception, dialog systems, or some combination thereof.

NLP algorithms are typically based on machine learning algorithms. Instead of hand-coding large sets of rules, NLP can rely on machine learning to automatically learn these rules by analyzing a set of examples (i.e. a large corpus, like a book, down to a collection of sentences), and making a statistical inference.

In general, the more data analyzed, the more accurate the model will be. NLP algorithms are mostly used for the following tasks:

- **Summarize blocks of text** using *Summarizer* to extract the most important and central ideas while ignoring irrelevant information.
- **Automatically generate keyword tags** from content using *AutoTag*, which leverages LDA, a technique that discovers topics contained within a body of text.
- **Identify the type of entity extracted**, such as it being a person, place, or organization using Named Entity Recognition.
- Use *Sentiment Analysis* to **identify the sentiment of a string of text**, from very negative to neutral to very positive.
- **Reduce words to their root**, or stem, using *PorterStemmer*, or **break up text into tokens** using *Tokenizer*.

## Open Source NLP Libraries

These libraries provide the algorithmic building blocks of NLP in real-world applications. *Algorithmia* provides a free API endpoint for many of these algorithms, without ever having to setup or provision servers and infrastructure.

- **Apache OpenNLP**: a machine learning toolkit that provides *tokenizers*, sentence segmentation, *part-of-speech tagging*, *named entity extraction*, *chunking*, *parsing*, co-reference resolution, and more.
- **Natural Language Toolkit (NLTK)**: a Python library that provides modules for processing text, classifying, tokenizing, stemming, tagging, parsing, and more. **In our project, we have chosen NLTK library for our work.**
- **Stanford NLP**: a suite of NLP tools that provide *part-of-speech tagging*, *the named entity recognizer*, *co-reference resolution system*, *sentiment analysis*, and more.
- **MALLET**: a Java package that provides Latent Dirichlet Allocation, document classification, clustering, topic modeling, information extraction, and more.

### **WHAT IS A JOURNAL?**

A journal is a detailed account that records all the financial transactions of a business, so that they can then be used for future reconciling and transfer to other official accounting records, such as the general ledger.

A journal states the date of a transaction, which accounts were affected and the amounts, usually in a double-entry bookkeeping method.

### **WHAT IS A LEDGER?**

A general ledger is a company's set of numbered accounts for its accounting records. The ledger provides a complete record of financial transactions over the life of the company. The ledger holds account information that is needed to prepare financial statements and includes accounts for assets, liabilities, owners' equity, revenues and expenses.

### **MOTIVATION BEHIND SOFTWARE-DRIVEN RECORDKEEPING**

- As mentioned in the previous section that ledger and journal entries are of prime importance to any economic body. Such a pivotal facet should always be implemented under the accurate supervision of a software-oriented system.
- And not just the recordkeeping but nobody would want to waste much time doing all the hard work to find the accounts involved, the accounts credited etc. and our application relieves the user from all such pain and accepts any query in the same way he would “speak” them to any other person under the sun!
- The user just feed the query and the rest is done by the application.
- The user may view the status of the entries at any point of time and this will be done by querying into the local database.

# PROJECT

**AIM:** Our project is aimed at building a software application that allows users to store and retrieve their complex day-to-day monetary transactions into a well-defined accounting database (journal entries, ledger accounts, etc.) with a broad scope that accommodates almost all types of transactions.

**LIBRARIES USED:** NLTK python library.

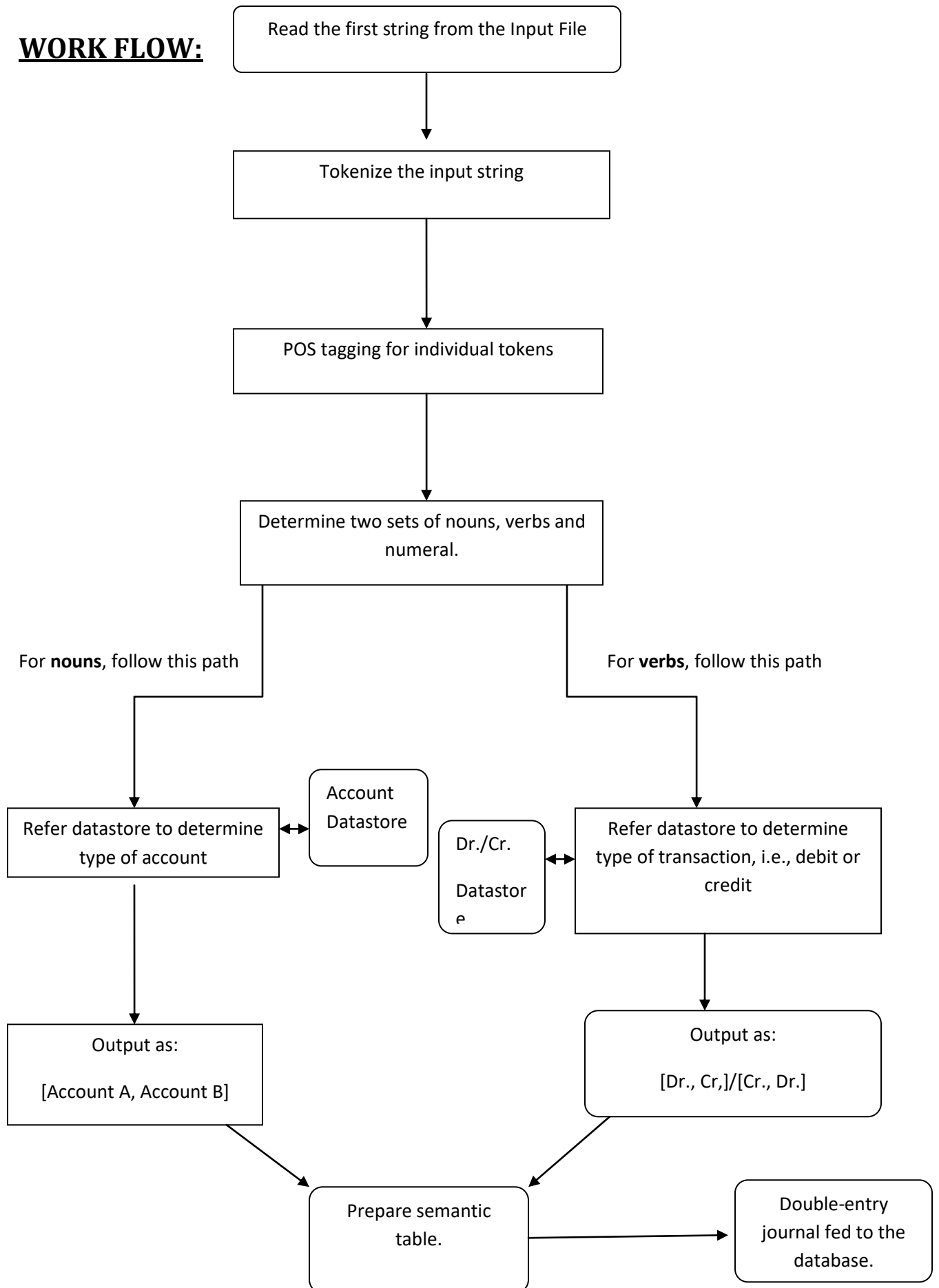
## **PROCEDURE:**

- First we take an **input text file** which consists of all the transactions made.
- Then, we **extract** each sentence (here sentence indicates a single COMPLETE transaction).
- Using the **NLTK toolkit**, we convert each sentence into tokens, using the **nltk.word\_tokenize ()** method.
- Once the sentence is converted into a set of tokens, we can easily **classify** them as to the important words that are needed to determine whether a particular transaction will be in the debit section or the credit section.
- To make the classification easier, we use the **pos\_tag\_sents()** in the NLTK library to understand as to what *part-of-speech* does the given token belong to.
- For a double-entry journal, we divide the transactions into two sections, each containing a set of consecutive **nouns**, **verbs** and **a numeral**. For each section repeat the following steps:

- The first set of nouns is fed to an algorithm that determines which **type of account** it represents. For this, we have a database that stores the synonymous words for each type of account.
- Similarly, the first set of verbs is fed to an algorithm to determine the type of transaction (**debit** or **credit**). For this, we have a database that stores the synonymous words used in **debit** or **credit**.
- The **numeral** determines the **amount** associated with the transaction.
- Once the classification is made, we build our own keyword directory, i.e. a **semantic table** that contains a list of keywords which helped us define the journal entry of the given transaction.
- The journal entry for each transaction is represented in the following format, and is fed into the database.

Date	Particulars		Amount
	Dr.	Cr.	(Rs.)

## WORK FLOW:





## **EXAMPLE:**

As an example let us have a transaction that says

“Jan 2:        An amount of \$36,000 was paid as advance rent for three months.”

Here, the sentence is converted into set of tokens:

[('Jan', 'NNP'), ('2', 'CD'), ('An', 'DT'), ('amount', 'NN'), ('of', 'IN'), ('\$', '\$'), ('36,000', 'CD'), ('was', 'VBD'), ('paid', 'VBN'), ('as', 'IN'), ('advance', 'NN'), ('rent', 'NN'), ('for', 'IN'), ('three', 'CD'), ('months', 'NNS'), ('.', '.')] ]

1<sup>st</sup> section:

Noun: amount                      ->    Cash Account

Verb: was paid                      ->    Credit

Numeral: 36,000

2<sup>nd</sup> section:

Noun: advance rent                ->    Advance Rent Account

Verb:                                      ->    Debit

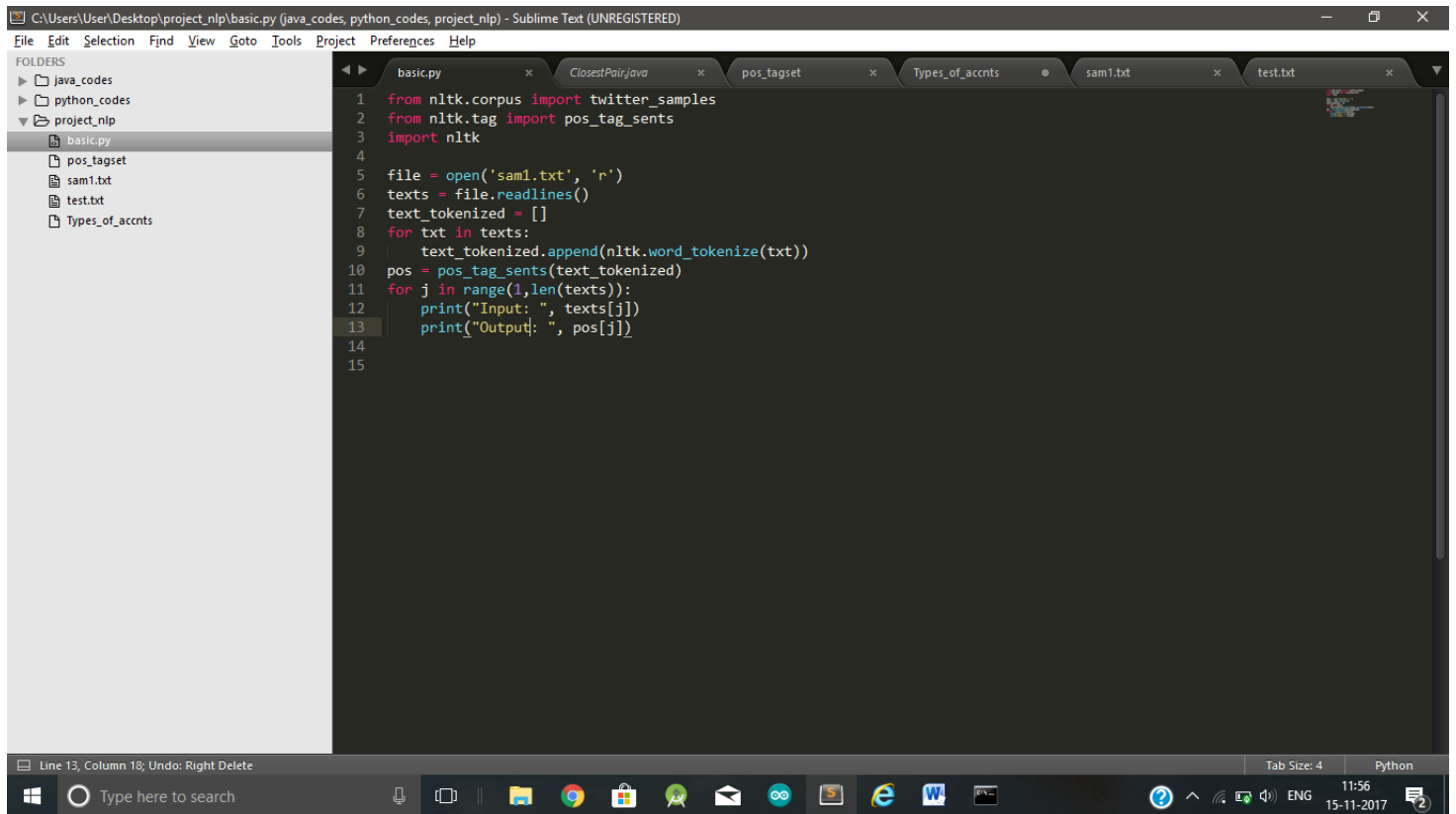
Numeral:

The corresponding journal entry:

Jan 2	Advance Rent A/C	36000
	Cash A/c	36000

---

## CODE:



The screenshot shows a Sublime Text editor window with the title bar "C:\Users\User\Desktop\project\_nlp\basic.py (java\_codes, python\_codes, project\_nlp) - Sublime Text (UNREGISTERED)". The left sidebar displays a file explorer with the following structure:

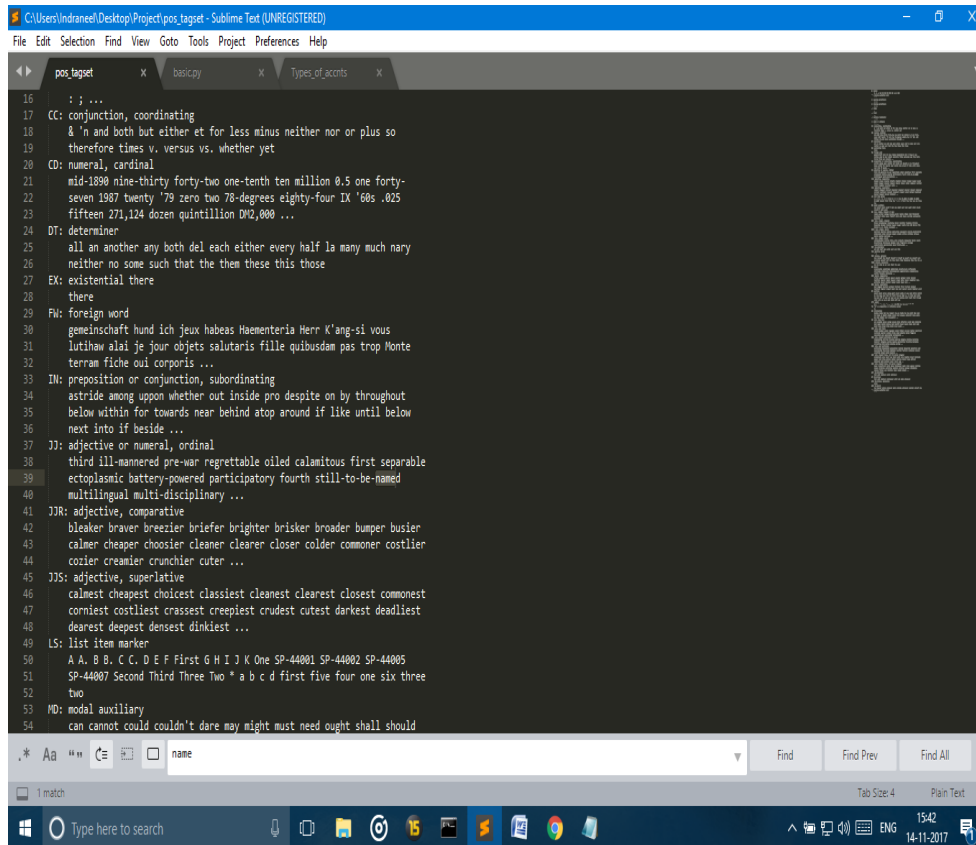
- FOLDERS
  - java\_codes
  - python\_codes
  - project\_nlp
    - basic.py
    - pos\_tagset
    - sam1.txt
    - test.txt
    - Types\_of\_accnts

The main editor area shows the code in `basic.py`:

```
1 from nltk.corpus import twitter_samples
2 from nltk.tag import pos_tag_sents
3 import nltk
4
5 file = open('sam1.txt', 'r')
6 texts = file.readlines()
7 text_tokenized = []
8 for txt in texts:
9     text_tokenized.append(nltk.word_tokenize(txt))
10 pos = pos_tag_sents(text_tokenized)
11 for j in range(1, len(texts)):
12     print("Input: ", texts[j])
13     print("Output: ", pos[j])
14
15
```

The status bar at the bottom indicates "Line 13, Column 18; Undo; Right Delete", "Tab Size: 4", and "Python". The Windows taskbar at the very bottom shows the search bar and various application icons.

- As elaborated above, the python code uses NLTK library to tokenize the string and then do the Part-of-Speech tagging to each and every keyword present in the string.



- A basic part-of-speech tagging chart that shows how each keyword is mapped to its corresponding POS.

# OUTPUT:

```
Command Prompt
Microsoft Windows [Version 10.0.15063]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\User>cd Desktop

C:\Users\User\Desktop>cd project_nlp

C:\Users\User\Desktop\project_nlp>python basic.py
Input: Jan 2 An amount of $36,000 was paid as advance rent for three months.

Output: [('Jan', 'NNP'), ('2', 'CD'), ('An', 'DT'), ('amount', 'NN'), ('of', 'IN'), ('$', '$'), ('36,000', 'CD'), ('was', 'VBD'), ('paid', 'VBN'), ('as', 'IN'), ('advance', 'NN'), ('rent', 'NN'), ('for', 'IN'), ('three', 'CD'), ('months', 'NNS'), ('.', '.')]
Input: Jan 13 Provided services to its customers and received $28,500 in cash.

Output: [('Jan', 'NNP'), ('13', 'CD'), ('Provided', 'NNP'), ('services', 'NNS'), ('to', 'TO'), ('its', 'PRP$'), ('customers', 'NNS'), ('and', 'CC'), ('received', 'VBD'), ('$', '$'), ('28,500', 'CD'), ('in', 'IN'), ('cash', 'NN'), ('.', '.')]
Input: Jan 14 Paid wages to its employees for first two weeks of January, aggregating $19,100.

Output: [('Jan', 'NNP'), ('14', 'CD'), ('Paid', 'NNP'), ('wages', 'NNS'), ('to', 'TO'), ('its', 'PRP$'), ('employees', 'NNS'), ('for', 'IN'), ('first', 'JJ'), ('two', 'CD'), ('weeks', 'NNS'), ('of', 'IN'), ('January', 'NNP'), ('.', '.'), ('aggregating', 'VBG'), ('$', '$'), ('19,100', 'CD'), ('.', '.')]
Input: Jan 25 Received $4,000 as an advance payment from customers.

Output: [('Jan', 'NNP'), ('25', 'CD'), ('Received', 'VBD'), ('$', '$'), ('4,000', 'CD'), ('as', 'IN'), ('an', 'DT'), ('advance', 'JJ'), ('payment', 'NN'), ('from', 'IN'), ('customers', 'NNS'), ('.', '.')]
Input: Jan 26 Purchased office supplies costing $5,200 on account.

Output: [('Jan', 'NNP'), ('26', 'CD'), ('Purchased', 'VBD'), ('office', 'NN'), ('supplies', 'NNS'), ('costing', 'VBG'), ('$', '$'), ('5,200', 'CD'), ('on', 'IN'), ('account', 'NN'), ('.', '.')]
Input: Jan 28 Paid wages to its employees for the third and fourth week of January: $19,100.

Output: [('Jan', 'NNP'), ('28', 'CD'), ('Paid', 'NNP'), ('wages', 'NNS'), ('to', 'TO'), ('its', 'PRP$'), ('employees', 'NNS'), ('for', 'IN'), ('the', 'DT'), ('third', 'JJ'), ('and', 'CC'), ('fourth', 'JJ'), ('week', 'NN'), ('of', 'IN'), ('January', 'NNP'), (':', ':'), ('$', '$'), ('19,100', 'CD'), ('.', '.')]
Input: Jan 31 Paid $5,000 as dividends.

Output: [('Jan', 'NNP'), ('31', 'CD'), ('Paid', 'NNP'), ('$', '$'), ('5,000', 'CD'), ('as', 'IN'), ('dividends', 'NNS'), ('.', '.')]
Input: Jan 31 Received electricity bill of $2,470.

Output: [('Jan', 'NNP'), ('31', 'CD'), ('Received', 'NNP'), ('electricity', 'NN'), ('bill', 'NN'), ('of', 'IN'), ('$', '$'), ('2,470', 'CD'), ('.', '.')]

C:\Users\User\Desktop\project_nlp>
```

## **AREAS TO WORK ON**

1. THE CLI ENVIRONMENT FOR THE USERS:
2. THE DATASTORE TO SUPPORT SELF AUTOMATION
3. SEPARATING USER ACCOUNTS TO ALLEVIATE CHANCES OF ENTRY INCONSISTENCY AND HELP ADMIN AUTOMATION
4. FULL TESTING OF THE APPLICATION