Pilot Operating Handbook Semantic Text Analysis

Project Solution Approach

**The Boeing Company**

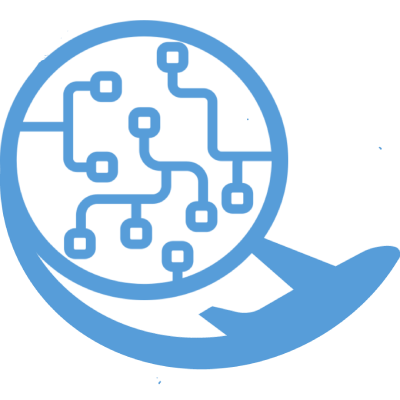


**The Autopilots**

Katie Peterson

Allison Lorphanpaibul

Weiren Lai

Daniel Harker

**TABLE OF CONTENTS**

[**Introduction**](#_heading=h.uho958i4tuz3) **2**

[**System Overview**](#_heading=h.3znysh7) **2**

[**Architecture Design**](#_heading=h.2et92p0) **2**

[Overview](#_heading=h.tyjcwt) 2

[Subsystem Decomposition - Daniel](#_heading=h.3dy6vkm) 3

[*Sentence Parser*](#_heading=h.1t3h5sf) *5*

[Description](#_heading=h.4im6k8py312n) 5

[The sentence parser will be used to grab every sentence from the input files. It is simply reading all of the text from the file and tokenizing it by sentence.](#_heading=h.q06jhrowdf5p) 5

[Concepts and Algorithms Generated](#_heading=h.69w5sfyj56d6) 5

[In order to parse through by sentence, the program will just need to look for a stop character.](#_heading=h.sxt9xes758mf) 5

[Interface Description](#_heading=h.u814d1eeogvh) 5

[*Noun Parser*](#_heading=h.4d34og8) *5*

[Interface Description](#_heading=h.eb25nqa6va04) 5

[Database](#_heading=h.cnhx615eqrrh) 6

[**Data design - Katie**](#_heading=h.2s8eyo1) **6**

[**User Interface Design**](#_heading=h.17dp8vu) **9**

[**Summary of the State of This Project**](#_heading=h.3rdcrjn) **11**

[**Future Work for This Semester**](#_heading=h.26in1rg) **11**

[**Glossary**](#_heading=h.lnxbz9) **11**

[**References**](#_heading=h.35nkun2) **11**

[**X. Appendices**](#_heading=h.1ksv4uv) **11**

# Introduction

Artificial intelligence (AI) and machine learning (ML) have become the trends of the future; however, the cost of AI and ML is very high, especially in acquiring and maintaining real and production data. Hence, there is a need to be able to support training synthetic data engines like GANs (generative adversarial networks) to create synthetic data on-demand. These engines provide a fast and inexpensive way to produce data that is properly marked and tagged, unlike most real and production data.

Our goal is to develop a semantic text parsing algorithm that takes vocabulary from Boeing aircraft handbooks and stores each noun/noun phrase with its designated document name, year, product, location, and 1-50 sentences to demonstrate context for each noun. We expect the program to have a 60-90% quality after training the GAN [1].

# System Overview

This project’s goal is to develop a semantic text parsing algorithm that takes vocabulary from Boeing aircraft handbooks and stores each noun/noun phrase with its designated document name, year, product, location, and 1-50 sentences to demonstrate context for each noun. By achieving that, we divided the project into three sections of design Architecture, Data and UI design.

In architecture design, the system will use a simple data pipeline architecture. From raw data to an output through internal subsystems, which contain three major systems: data source for the input, NLP for the process, and database for the output.

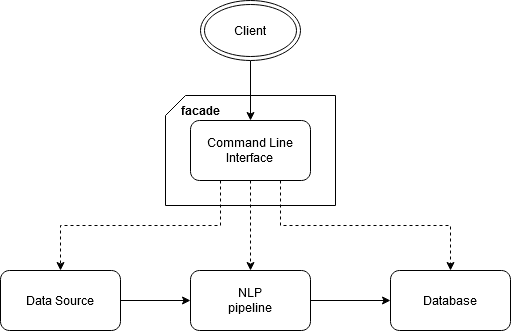
In data design, we found that using MySQL will be the most suitable database in terms of a relational database by storing all the information of the noun/noun phrase.

In UI design, since the goal isn’t to be fancy, we have decided to use command lines at first to import files, start processing, and finish. However, there could be a great extent in terms of UI with JS in the future if we have enough time.

# Architecture Design

## Overview

The system will use a simple data pipeline architecture. Raw data from the user will be input at the start of the pipeline, processed by the internal subsystems, and finally be output to a database.



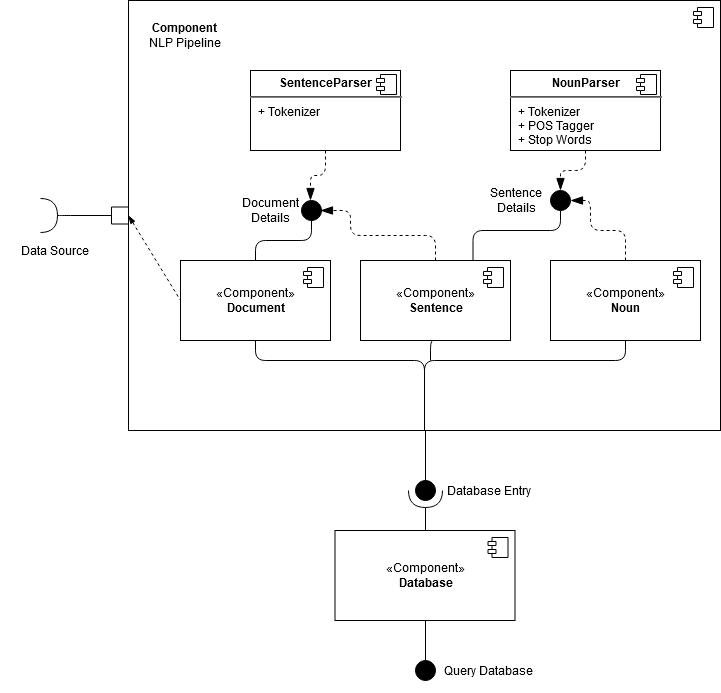
Since our system does not require that much user interaction, the interface layer of the system will be simple. The *facade* design pattern can be employed here to provide a simple way for the user to interact with the system easily, without having to view all the processes under the hood, reducing complexity. To start, we can use the command line as an interface for the user. This pattern will also provide room for extensibility of the interface in the future if there is time as described in [2].

At this time, our system will not require a lot of information hiding, so the interface does not have to be too sophisticated.

## Subsystem Decomposition - Daniel

We identified three major systems for our project: the data source for input, the NLP pipeline for processing, and the database for output. There will also be a basic UI layer for getting command-line input from the user. This way of breaking up the system will let each system be relatively independent of each other, besides providing interfaces for the other systems to communicate with it.

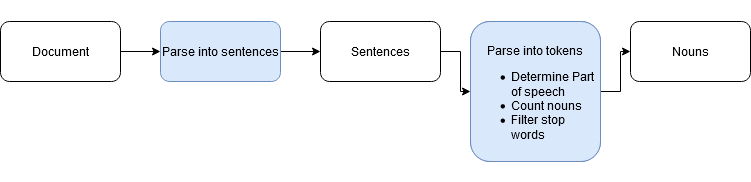
Below is a possible diagram of the system components, with each component representing a subsystem:



Since the natural language processing pipeline is the most crucial and largest part of the system, breaking it up further into subsystems was important. Within the pipeline, there will be three internal structures for storing the data at different stages of processing-- the total document, sentences in the document, and nouns in the document. Because of the nature of the data, these structures will need to be highly dependent on each other. Since the database will follow a similar structure with tables as the internal structures, this decomposition will make handling the output data more streamlined. Handling the input and output will most likely be relatively straight-forward, with them both requiring interfaces for the pipeline and user to interact with.

Besides the data structures, the pipeline will also contain parsers that will act as interfaces between the tightly-related data structures. The SentenceParser will be just for separating the document into sentences. These sentences will be important for providing context for the nouns. The other parser, the NounParser will extract nouns from sentences. This is also where we will apply other NLP algorithms, such as stop words and lemmatization.

Below is a flowchart for the data processing pipeline:



Below is a more detailed description of the different subsystems:

### Sentence Parser

#### Description

#### The sentence parser will be used to grab every sentence from the input files. It is simply reading all of the text from the file and tokenizing it by sentence.

#### Concepts and Algorithms Generated

#### In order to parse through by sentence, the program will just need to look for a stop character.

#### Interface Description

The program will simply open the file for reading, then tokenize the text in the file into sentences. The end of sentences will be determined by a stop character, which in this case will be a period. Each sentence will then be added to the database with a unique ID.

### Noun Parser

1. ***Description***

In addition to parsing through the text all getting every sentence, the same thing will need to be done with all of the nouns. The nouns are the crucial aspect when using the GAN to create sentences.

1. ***Concepts and Algorithms Generated***

After compiling all sentences using the sentence parser, the next step will be to parse each individual sentence to look for nouns. This will be done using parts of speech tagging which determines what a word is (i.e. noun, verb, adjective, etc.) and selecting only the nouns. Nouns will be counted to accommodate repeats. Nouns will also have links to the sentences they came from.

#### Interface Description

After completion of text parsing and storing all sentences in the database, similar actions will be done with the nouns. Each sentence will be parsed by word, and each word will be analyzed using parts of speech tagging. Every word found to be a noun will be stored in the database. New nouns will be given a unique ID and given a count of 1; repeated nouns will simply increment the count. For each noun in the current sentence, they will be linked to that sentence’s ID for context.

### I.1.3. Database

1. ***Description***

The database will be used to store both the sentences from the input files as well as all of the nouns that have been identified in each sentence.

1. ***Concepts and Algorithms Generated***

The original plan was to store only the nouns from the input file, and not have any sort of information stored for the sentences. It was later determined that having the context of the noun will help the GAN create sentences and increase the accuracy of our tests.

1. ***Interface Description***

For details regarding database design, see section IV below.

Services Provided:

1. Service name: Get text from file

Service provided to: Sentence Parser

Description: Reads in text from the input files provided by Boeing. The input will be tokenized by sentence and stored in a database.

2. Service name: Get nouns from sentence database

Service provided to: Noun Parser

Description: Reads in sentences from the sentence database. The input will be tokenized by nouns using a parts of speech analyzer and stored in another, linked database.

Services Required:

This essentially boils down to getting the nouns, with context, from the input files provided by Boeing. It requires the use of the Sentence Parser and Noun Parser to tokenize the input text from hundreds of pages to sentences and finally to nouns.

# Data design - Katie

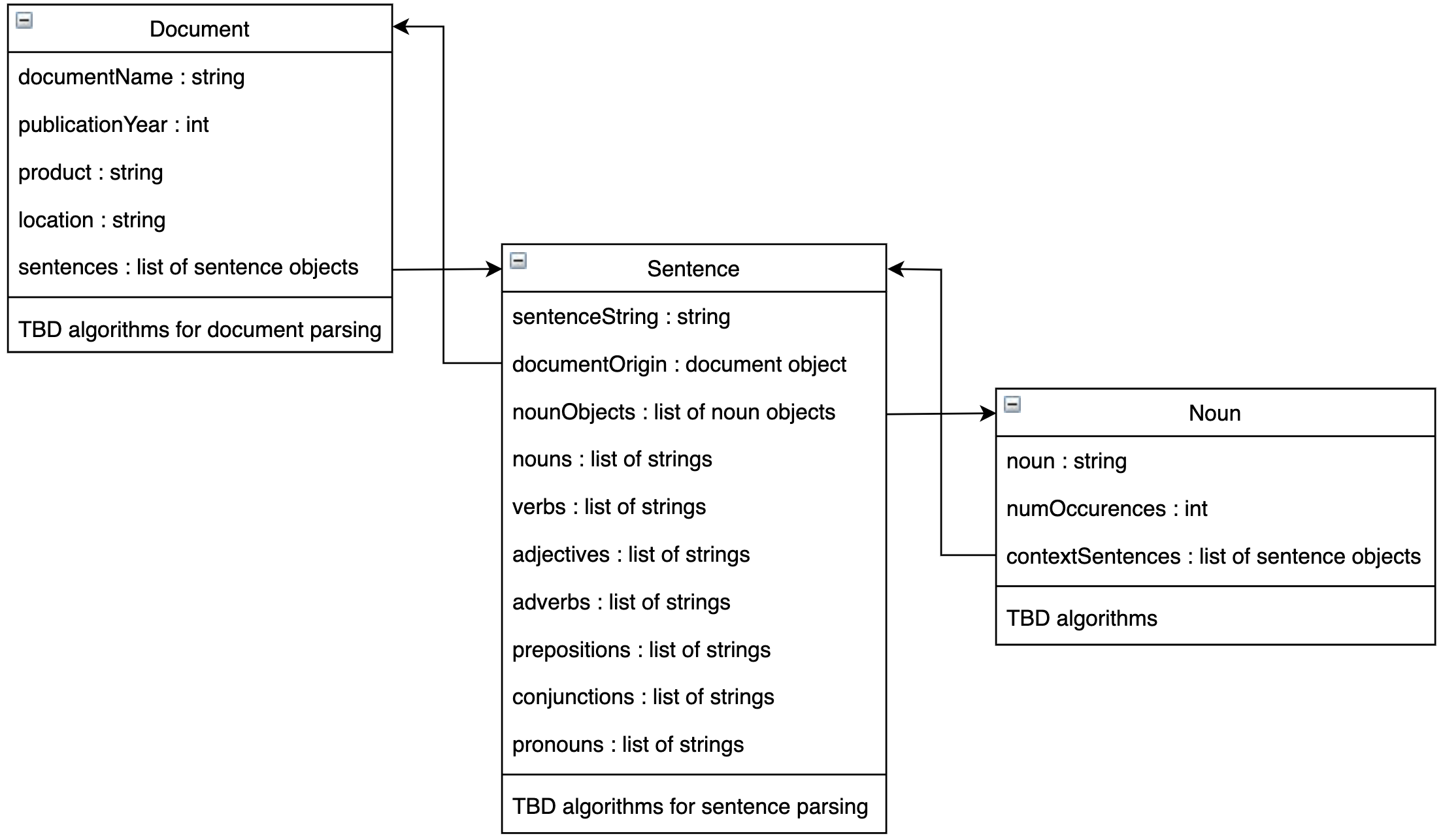
**Internal Data Structures**

There will be classes and objects within the Python program utilized to temporarily store data. Specifically, there will be a document class, a sentence class, and a noun class.

The document class will consist of properties that store a document name, its publication year, its file location, the product the document describes, and a list of sentence objects - one for each sentence in the document. It will also contain methods that when run, will search the document and assign values to the properties. A document object will be created for each document read as input to the system.

The sentence class will consist of properties that store the full sentence, the object of the document it appears in, a list of noun objects - one for each noun appearing in the sentence, and various lists of strings for the different parts of speech. It will also contain methods for sentence parsing that will assign values to the properties. A sentence object will be created for each sentence identified in a document.

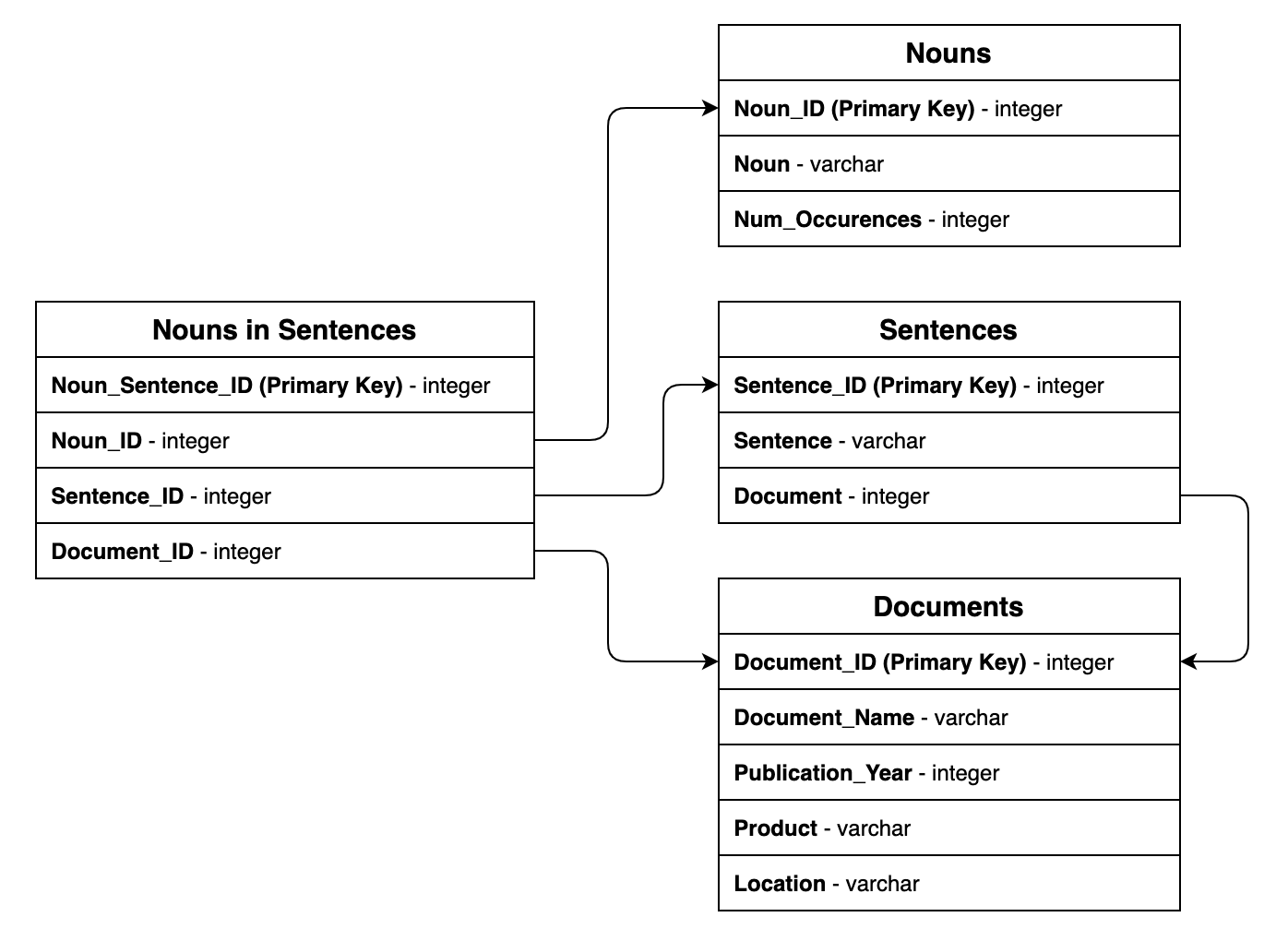
The noun class will consist of properties that store the noun or noun phrase, the total number of times the system has encountered the noun, and a list of sentence objects - one for each sentence the noun appears in. It will also contain methods to assign values to the properties. A noun object will be created for each unique noun the system encounters.



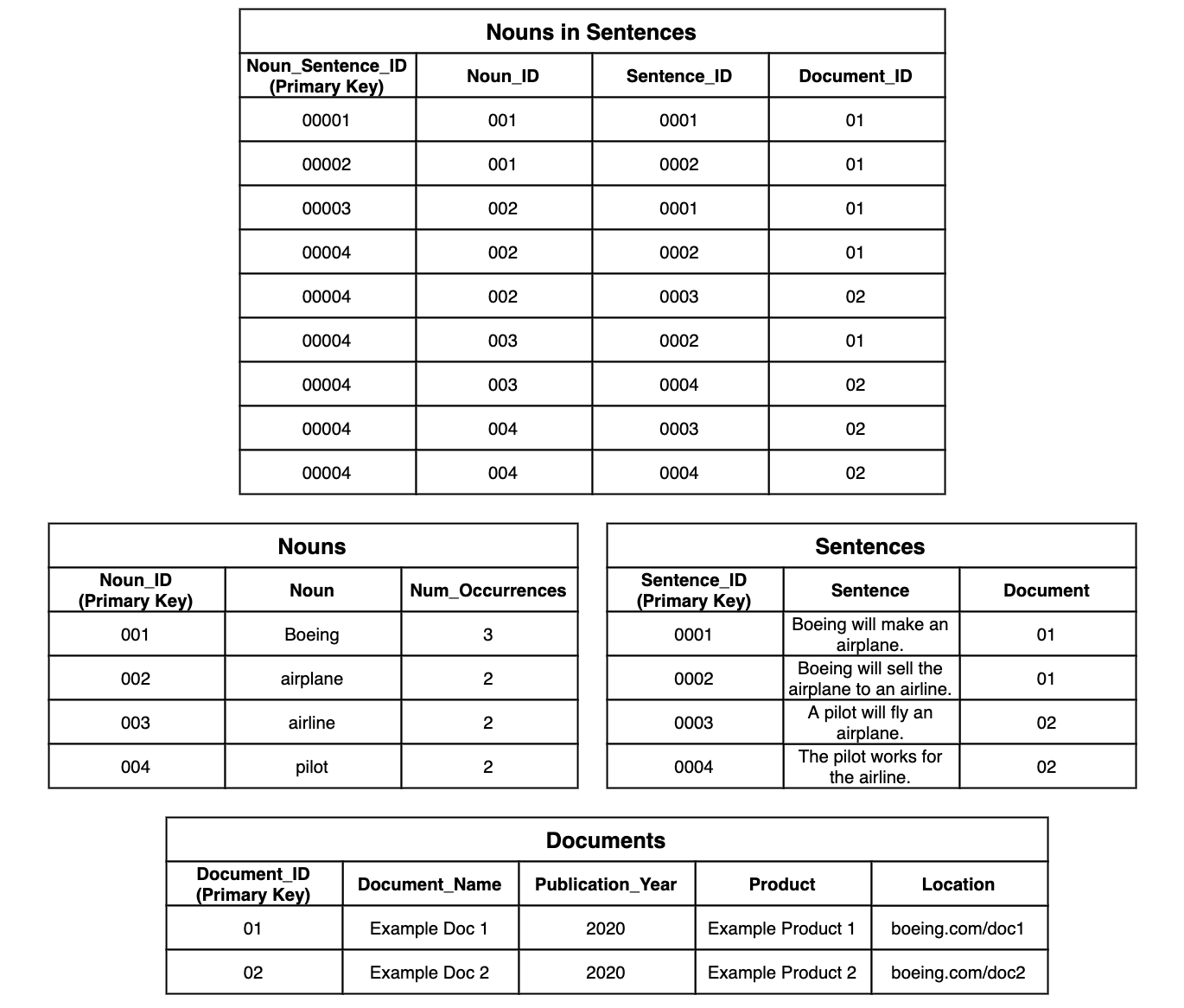
**MySQL Database**

Our system will export the data it collects to a MySQL relational database. The database will consist of four tables. The first table will contain the name of all documents used as input to the system, along with their publication year, product they were written for, and the location where they can be found. The second will contain a list of all sentences read by the system, along with the document they appear in. The third table will list all unique nouns found by the system and the total number of times they appear in the documents. Finally, the fourth table will relate the documents, sentences, and nouns by listing each noun’s primary key with the primary key of each sentence it appears in, along with the primary key of the document that sentence appears in.

Below is a visualization of the database and its 4 tables:



Below is a simple example of the database after the system takes two documents as input:



# User Interface Design

Based on our client’s expectation we’ve developed a rough idea of how the user interface v1.0 is going to be – CMD.

At first, due to the fact that we are importing Boeing handbooks and the language we are using, we need to import text files in order to begin with our program. With a prompt to the user: would you like to import a file? There are a fews ways of importing files, read(), readline() and readlins(), before importing the files, we would want to prompt user for the file which we could use tkinter to do so, it will pop out a window and user could choose which filename to import. However, we will determine which to use based on our need because each function reads in differently, which might have an impact in terms of storage.



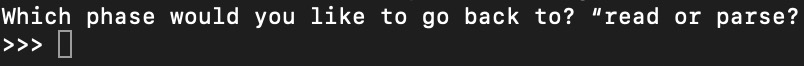
After reading in the text, we parse the Boeing handbook into sentences, which we also thought that would be a prompt to the user before parsing the file. Hence, we would prompt the user with a simple prompt: Would you like to parse the file? Y/N, if the user answers “Y” it will start parsing, if the user says “N” it will take the user back to the first prompt.

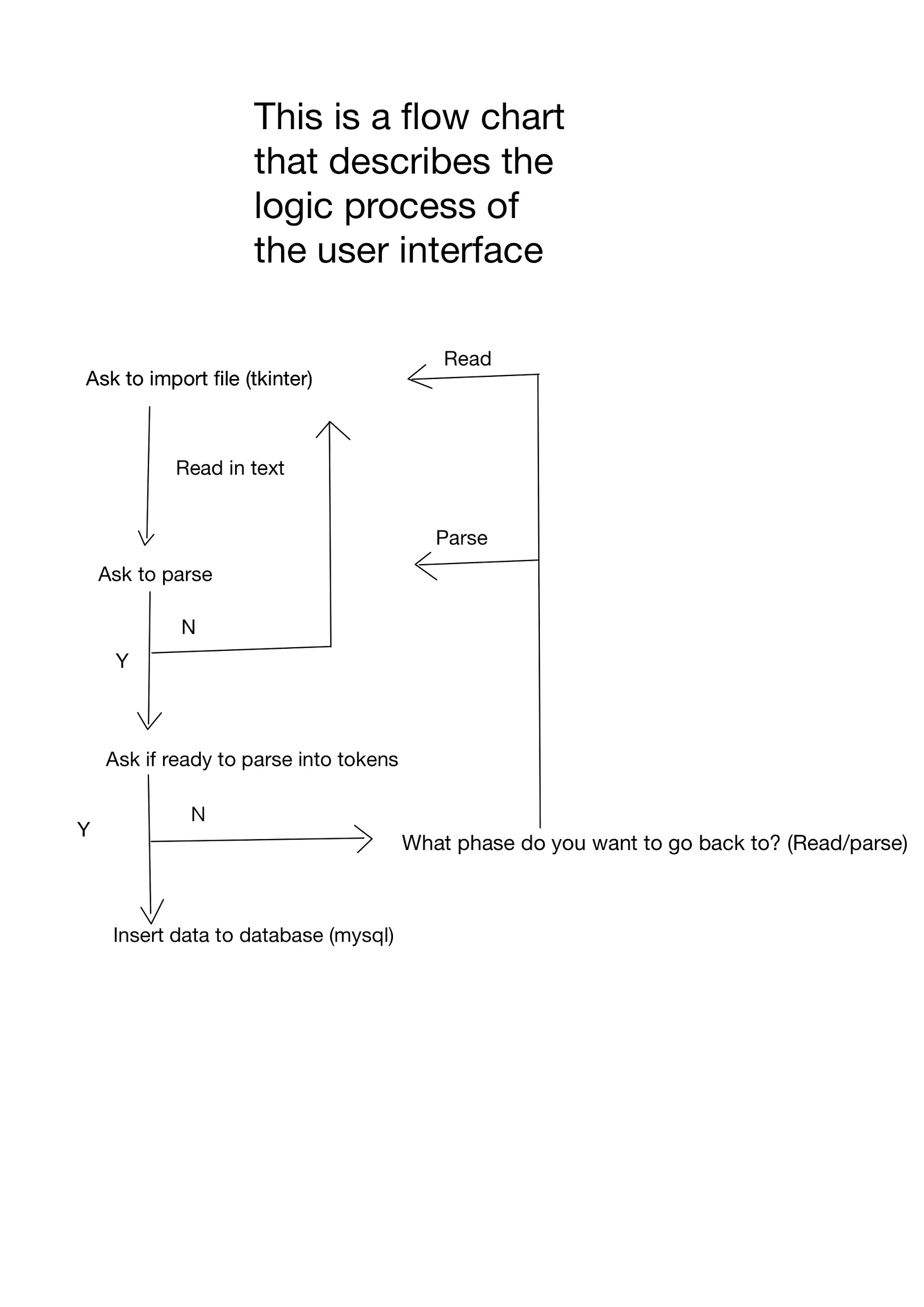


When the sentences are parsed and stored into our database, we would like to prompt the user for the next move - parse into tokens by asking: Ready to parse into tokens? Y/N? If the user answers “Y”, it will start parsing, if the answer is “N”, it will prompt the user with a question: Which phase would you like to go back to? “read or parse?”. When the user chooses to read, it will start from the beginning: asking the user to import a file. When the user chooses parse, it will take the user back to the parsing phrase.









# Summary of the State of This Project

So far we have completed a rough draft of how we are going to do in the next few weeks by having a Gantt chart. We have been treating program architecture design, database design along with UI design with extreme care because they will have a huge impact in terms of future working and implementing. We are now ready to move on to implement our designs in the real application.

# Future Work for This Semester

We used the following Gantt chart to organize the project timeline:

<https://docs.google.com/spreadsheets/d/1lJD5gY3dHMCfuv3KDJAOWPmw1VpoPJ7pAH-iBIoO5Ts/edit?usp=sharing>

We are using extra milestones to make sure we deliver the minimum viable product on time. Work for the prototype was separated between the data processing and data output steps. People will own different aspects of the project. Since some parts of the system will need to talk to each other, the people owning those respective parts will work together as well.

# Glossary

**Generative Adversarial Networks (GANs) -**  A popular machine learning model used to create synthetic data on-demand.

**Natural Language Processing (NLP)** - The field pertaining to how to program machines to

understand human language.

**Simplified English -** The standard for written aerospace documents. Utilizes a restricted vocabulary and simple grammar rules.

# References

[1] (Dutoit, 2010), 3rd Edition, by Bernd Bruegge and Allen H. Dutoit, Prentice Hall, 2010.

[2] Gamma, Erich, *Design Patterns: Elements of Reusable Object-Oriented Software.* Reading, Massachusetts, USA: Addison-Wesley Publishing Company, 1995. [Online]. Available. <https://archive.org/details/dehttps://archive.org/details/designpatternsel00gamm/page/185/mode/2upsignpatternsel00gamm/page/185/mode/2up>. Accessed: October 23, 2020.

# X. Appendices

Any larger images, charts, or external materials should be put into appendices. These are attached at the end of the document so the main materials are kept closer together and the overall flow of the document is preserved. If you include 4 pages of spreadsheets in the middle of a section, it makes it very difficult to track the flow of your presentation. Instead, those sheets go in Appendix [X] and are referred to by the earlier document.

You may have as many appendices as you need for the document to make sense.