Design Lightning Talk

William Sengstock, Kelly Jacobson, Zach Witte, Austin Buller, Sam Moore, Dan Vasudevan, Jacob Kinser

3.1.1 Broader Context

Area	Concerns
Public health, safety, and welfare	client is primary stakeholder, building on knowledge base
Global, cultural, and social	bias in software development
Environmental	electricity to power our own devices
Economic	none

3.1.2 User Needs

- A student working on a programming assignment needs to find the right Java documentation related to their project because they want to understand how a piece of code works.
- The lead developer on a team of programmers needs to analyze the code written by their team because they want insight into how their team approaches coding.
- A new hire needs to understand how an outdated program works because they are tasked with updating and debugging it.
- These user groups need some functionality that lets them more easily look at code and the documentation of code in order to understand something about that code.

3.1.3 Prior Work/Solutions

- IEEE Paper “Choosing an NLP Library for Analyzing Software Documentation: A Systematic Literature Review and a Series of Experiments”
- Advantage: we are doing a similar thing so we can use paper as a guide
- Disadvantage: the research in the paper uses different models, has their own comparison tools, does not modify models
- Experts vs. Amateurs

3.1.4 Technical Complexity

- Learning NLP for the first time
- Many different NLP models and many NLP techniques and functions

- Our project is based on current industry standards and will expand on them

3.2.1 Design Decisions

- NLP Toolkit
- Word Embedding Technique
- Method of Training for NLP Model

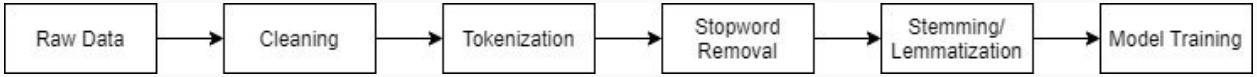
3.2.2 Ideation

- Which NLP Toolkit would best fit our needs?
  - NLTK
  - SpaCy
  - StandfordNLP
  - Text Blob
  - Apache
- Client gave recommendations
- Look at other NLP models and what worked

3.2.3 Decision-Making and Trade-Off

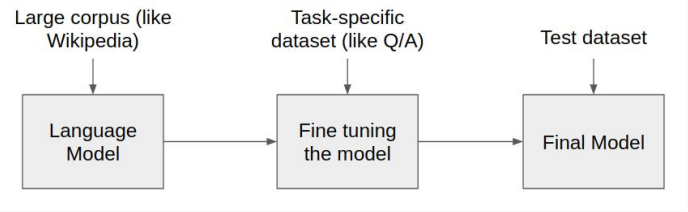
- The best way to learn is through experience
- Try to create our own models with different data sets
  - What is easy to use and understand?
  - What provides functions that will be useful for analyzing software documentation?
  - Which model was the most accurate and provided expected output?
- Experiment and share findings

3.3.1 Design Visual and Description



- Raw Data: For the purpose of this project, the raw data will consist of software documentation.
- Cleaning: All non-letters, such as punctuation, is removed from the data. Makes sure only words are left.
- Tokenization: Process of splitting the data up into tokens, can be divided as whole words, characters, or subwords. Allow us to evaluate the data.
- Stop Word Removal: Eliminating words that don't have much value, such as filler words. Examples of stop words are "a", "the", "are".
- Stemming and Lemmatization: Common methods for shortening words down to a common base. For example "running" would be cut down the the base word "run".
- Model Training: Taking in all of the above to train the model to make predictions based on the data given.

3.3.2 Functionality



Our design is specifically intended to operate in a software environment. NLP is not necessarily meant for software documentation, it has other various uses as well. For this project, our aim is to train these models to deal with taking in software documentation and then making predictions based off the data. Our model satisfies functional requirements like taking into account that it must output quantitative data so our team can understand how well it is working. It also satisfies non-functional requirements such as encouraging continuous improvement in choice of tokenization strategies.

3.3.3 Areas of Concern and Development

- Might not be able to implement a model that fully understands software documentation. Software documentation isn't written in traditional English, there are lots of abbreviations and commands. With there being a lot of different languages out there, it will be difficult to produce a flawless model.
- We will analyze models that have already been created for software documentation, and try to build off that to improve our model designs.