CECS Capstone Project
Group: Acato 1
Sage Gray
Nathan Ebbs
Cameron Kerley
Jacob Randles
Proposal Development
2/14/2025

Objectives

- Create a standard method of ingesting all expected document formats (PDF,Word,etc.).
- Choose an efficient data structure to store document information. This should be compatible with both suggested Natural Language Processing techniques(text embedding and bag of words).
- Use the chosen data structure and document parser to produce a dataset of documents and their assigned fitness class from the archive.
- Test the effectiveness of the 2 proposed methods against business leads with known classification not included in the dataset.
- Test both methods against new business leads and evaluate the results vs client expectations.
- Explore ways of combining both evaluation methods to improve accuracy and reduce false positives in the business leads.
- Make the final output a ranked number determining how compatible the solicitation is for the client and client company.

Potential Additional Milestones

- Create an automatic method to retrieve solicitations directly from trusted websites to process almost without human input.
- Make it able to automatically process and format documents into a readable and usable format.
- A short summary of the document output alongside the final ranking of compatibility.

Methodology

Method 1: Similarity Search

One approach to build the Rapid Identification and Vetting system(RIV) involves using modern AI technology for Natural Language Processing. This process uses a language model to take input text, like a document, and represent it as a vector that captures the semantics of that text. These vectors are called text embeddings and can be used to quantify the similarity between two pieces of text. This approach is well suited to the proposed task for its ability to utilize context and discern general similarities between differently phrased samples of text. One of the most common similarity metrics used for text analysis is cosine similarity. Effectively using this metric for the RIV will require making a dataset of proposals that are known good/bad matches for Acato's capabilities and embedding it with a language model. With this dataset incoming business leads can be embedded with the same model and checked for similarity between good and bad matches. Then we can use the overall similarity between groups to classify the lead as a good or bad match. This method can be extended to include varying levels of compatibility classification. The three current categories provided by the client are 'BadFit', 'GoodFit' and "GoodFitWithPartners". Further examination of the document archive may yield more categories of 'fitness' worth investigating to further refine the classification.

Method 2: Bag of Words

The second method we'll be trying is the Bag of Words machine learning model. This is a more simplistic approach that checks for words to which we give a positive or negative weight. While skimming a document, it will store weighted values of the words we've chosen and provide a sum or percentage that will show how close to our target documents the new document is. This is a far less computationally expensive method, but won't take into account the context of the words around it. This means we'll need to manually decide what synonyms of our positively and negatively weighted words may occur in future documents and have our program be on the lookout for them as well. We are planning to use this method as our starting point and move over to the Similarity Search method when we have a proof of concept. If our timeline runs longer than we expect, we will default to this simpler method because it will be easier to implement.

Suggested tools to build the Similarity Search system:

- Python:
    - The sentence-transformers module will provide an easy framework for interacting with models on Hugging-Face.
    - The sentence-transformers module also provides similarity metrics like cosine similarity.
    - Pandas dataframes can be used to store document embeddings with classifications.
    - Possible interface with the OpenAPI to summarize large documents before embedding.
    - Sklearn and scipy will be useful to attempt hierarchical clustering on archived documents.
    - A module like PyMuPDF will be useful for parsing documents.

Suggested tools to build the Bag of Words system:

- Python

    - NLTK (Natural Language Toolkit) or BeautifulSoup libraries to process words in a document
    - Pandas to store document embeddings in dataframes.
    - Sklearn and scipy will be used for feature extraction and CountVectorizer

Timeline

- Week 4: Create a document parser suitable for both methods.
- Week 5: Separate the document archive into training and test datasets using the document parser.
- Week 6: Start evaluating the dataset quality and begin implementing the methodologies described above.
- Week 7: Begin testing RIV identification against the test dataset.
- Week 8: Evaluate the test results against client requirements and identify weaknesses/bugs in both methods.
- Week 9-13: Iterate on the current system by fixing bugs and testing accuracy while receiving client and advisor feedback.