FHIS Project Plan

Description

The aim of this project is to build a proof-of-concept graded reader retrieval system to allow users to retrieve documents that are suitable for their reading level, but with some portion vocabulary outside that level so that they are challenged to improve their comprehension skills. The *ideal* system is envisioned as a multilingual web-based platform, with the capability to retrieve reading materials from the internet that match a user's CEFR (Common European Framework of Reference for Languages) level upon the user's request, with additional search parameters that allow the user to filter for specific grammatical constructions and cut-off thresholds to ensure only a specified fraction of vocabulary is from the user's CEFR level.

For the purposes of building a proof-of-concept system within 7 weeks, the project scope is limited to search through a locally-hosted collection of graded documents from the Spanish language, instead of a multilingual internet search engine. The priority of the project is to assemble a corpus of documents labelled with CEFR levels, then train a machine learning model on this corpus to learn to assign the reading level of any given document, build a document retrieval backend, and finish with a frontend GUI. Time permitting, the additional search parameters (grammar filters, vocabulary thresholds) may be incorporated.

Datasets

An ideal dataset for this project is a corpus of A- and B-level Spanish texts, with their CEFR level as labels. There is no restriction on the domain of the corpus. While there are many manually compiled collections of Spanish graded readers, there is currently no known corpus on the internet that suits the purpose of this project. Therefore, we will be building a corpus by scraping from available resources on the internet as well as from PDFs. We currently plan to scrape the following:

- Antología abierta de literatura hispana: An anthology of literature at the B1-B2 levels
- Lingua: Short texts with difficulty ranging from A1 to B2
- Spanish textbooks and literature in PDF format

One challenge with the data collection task is the lack of data. Gathering all resources, we expect to collect 10-20 documents for each of the A- and B-level, hence 30-40 documents in total. Although we will incorporate the use of linguistic features when training our model, it is possible that this amount of data will not be enough to develop a well-performing model. We will discuss with our mentor and the Capstone partners to arrive at a solution to improve the current situation.

Since all of the texts will be labelled with their CEFR levels, no further annotation is needed. When data collection is completed, this dataset will be stored in JSON format for easy distribution.

Expected deliverables

1. Assemble a balanced corpus of documents from different CEFR reading levels, labelled with their level.
2. Develop a locally-hosted web application that retrieves documents from a local corpus that match the user's specifications.

Functionality to incorporate into web application:

- *(optionally)* Self-diagnostic tool (or link to such a tool) to determine the user's CEFR reading level prior to performing search
- Specify user's CEFR reading level (options: A1, A2, B1, B2, C1, C2)
- Specify the percentage of vocabulary in the retrieved document that must be from that reading level
- *(time-permitting)* Additional search filters to specify whether certain grammatical constructions must be present or not
- Highlight words/phrases in the retrieved document(s) that are outside the user's reading level
- *(time-permitting)* Mouse-hover action to produce a live Spanish-to-English translation for selected words/phrases in the retrieved document(s)

Methods

Implementation and evaluation of the deliverables

- We will build a rule-based system as a baseline model. The rule-based system will use the vocabulary lists to classify the input text as level-A, level-B, or beyond.
- We will also build an SVM classifier which will assess readability of the input text based not only on the vocabulary lists but also on linguistic features extracted from the input text.

Methods for data handling

Once we assemble a corpus of A- and B-level documents, we will extract relevant features from the texts and store them in a dictionary form for each text.
We plan to use the following features for classification of texts:

- **Vocabulary lists (A, B, C):**
  We will build A-, B- and C-level vocabulary lists. These lists will be used by both the rule based system and the SVM classifier. Tokens in the input texts will be compared with the

vocabulary lists; the percentage of the tokens in the text in each level of vocabulary will be calculated and used as features in the classifier.

- **Frequency list:**
  We will also build a frequency list of types from the scraped texts. This list contains how frequently each word type appears in a fixed number of texts. This list will help to identify common words and rare words in the input text.

- **POS (Part-of-Speech):**
  We will tag part-of-speech for each token in the text. The POS tag will help us to determine the syntactic structure of the text. The POS tags and the syntactic structure of the text will be used as features in the classifiers.

- **Flesch reading ease:**
  We will use Flesch reading score to categorize texts on the level of reading difficulty: scores above 70 indicate text which are appropriate for primary school school students, scores between 70 and 50 indicate texts which can be read by secondary school students, and scores below 50 are appropriate only for college students.
  It is calculated using the following formula:
  $$206.835 - 1.015 \times \left(\frac{total\ words}{total\ sentences}\right) - 84.6 \times \left(\frac{total\ syllables}{total\ words}\right)$$

- **Syllabification:**
  We will try one of the following approaches to segment words into syllables:
  - tulengua.es
  - Building a segmentation model using a spanish dictionary with syllable information
  - Transfer learning (syllable segmentation) from English to Spanish
  - a rule-based model for syllable segmentation
  - morfessor Python package

- **The average length of sentences:**
  We will calculate the average length of sentences in the text. We expect that the readability of the text increases in proportion with the average length of sentences in the text.

- **Type-to-token ratio:**
  We will calculate the type-to-token ratio (TTR) for each text. A high value of TTR implies high lexical diversity while a low value of TTR implies low lexical diversity. We expect higher lexical diversity for more advanced texts.

- **Proportions of different POS tags (nouns, verbs, adjectives, prepositions, etc.):**
  We will count the proportion of different POS tags occurring within each text. We expect that there would be a correlation between the proportions of different POS tags and the

types of documents they are, and thus the difficulty level of the document. For example, government documents or scholarly papers tend to have a large proportion of nouns while narratives tend to have a large proportion of past verbs and third person pronouns. We would like to use this information to classify readability of texts.

- **The number of tokens, types:**
  We will use the total number of tokens and the total number of types in the text.

- **Number of sentences in the text:**
  We will use the total number of sentences in the text as a feature.

- **Other syntactic information:**
  We will use the counts of syntactic elements such as past aspect verb conjugations, present participle clauses, attributive adjectives, synthetic negation, etc., as features, since they indicate a more advanced reading level.

Packages/libraries that you will use

- spacy
- sklearn
- nltk
- tika (PDF reader)
- re (for regex)
- unicodedata (to convert special characters to ASCII for regex purposes)

Links to tutorials for training models or use existing models

- List of Spanish resources: https://github.com/dav009/awesome-spanish-nlp
- TextStat, can quickly calculate Flesch Reading Ease Score for Spanish (which we could use as one of our features): https://pypi.org/project/textstat/
- SVMS for NLP:
  https://monkeylearn.com/blog/introduction-to-support-vector-machines-svm/

Brief plan for testing code and conducting code reviews

- We will use test-driven development, testing as we write code, to make sure that any errors that appear are dealt with as soon as possible. The test-driven development cycle works as follows:

  1. Write at least one test,
  2. Implement just enough code to make it pass,
  3. Refactor the code so it is clean.

- We will use a version control repository (git) and use different branches in order to have the ability to modularize our code and roll back if needed.
- We will make sure to review each other's code before merging individual branches to the main branch.
- We will follow the principles of Continuous Integration by integrating our work frequently, and making sure that each integration is verified by automatic integration error detection (merge conflicts), and regularly testing to see that our code all works together. This will make sure that any integration errors are caught early on, and we will reduce integration problems in the future.
- We will make sure to write good quality, readable code with clear documentation so that our teammates can easily understand what our code does. We will also check that the code follows the DRY principle (Don't Repeat Yourself).

Schedule

Weekly update meeting with capstone partner: Mondays 11 a.m. PST
Weekly update meeting with mentor: Thursdays 11 a.m. PST

Week 1 (May 10 - May 14):
- Develop a baseline rule-based system (simple lexicon matching)
- Build a labelled training corpus
- Come up with linguistics features for feature extraction
- *(if time permits)* Get started on SVM

Week 2 (May 17 - May 21):
- Feature extraction
- Build SVM pipeline

Week 3 (May 24 - May 28):
- Finetune SVM
- Feature selection/addition
- *(if time permits)* Explore neural network approach

Week 4 (May 31 - June 4):
- Build interface for the final product
- Finalize (docker?)

Week 5 (June 7 - June 11):
- (To be determined)