Paulo Frazao
Xiaoyan Li
4 February 2020
COS Thesis Draft Paper

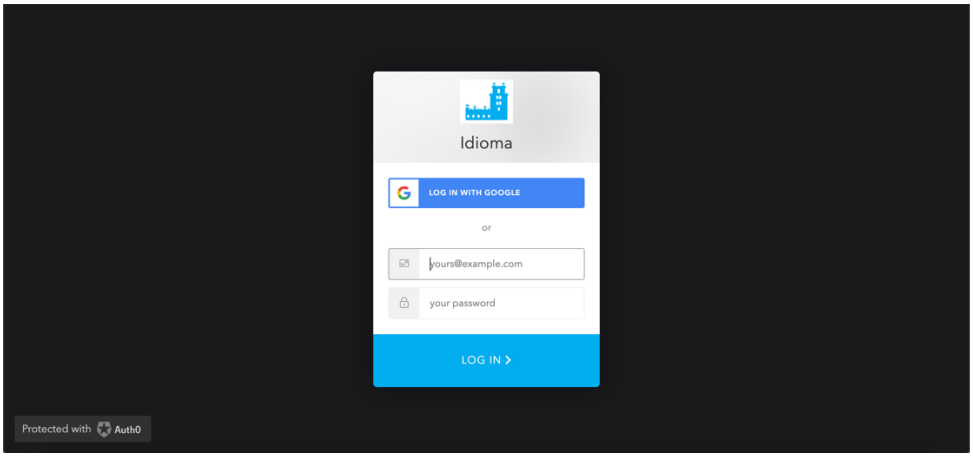
Thesis Outline:

- Chapter 1: Introduction
  - Problem Background
  - Motivation
  - Goal
    - Example Use Case
- Chapter 2: Related Work
  - Discuss related work at various depths, progressing from most broad to most specific
  - Layers of Related Work:
    - Language-learning applications (i.e. Duolingo, Memrise, etc.)
    - Systems/Applications that focus on reading comprehension development
    - Existing NLP / algorithmic techniques for specific modules of my application (i.e. tracking user progress, readability classification, dependencies amongst Portuguese grammar structures)
- Chapter 3: Approach
  - Description of approach: web-app, tracks user progress, adapts documents to user experiences, focus on reading comprehension fulfills a very specific niche and also helps prepare users for other kinds of learning (find stats for this)
  - Discuss layers of application + include diagram connecting front-end, back-end, database, models, scraper, etc.
- Chapter 4: Implementation
  - Explain every component in isolation:
    - Data
    - Web Scraper
    - Front-end + Back-end
    - Models
    - Database
    - Hosting (?)
  - Walk reader throw user flow -> various use cases that showcase each of the above components
- Chapter 5: Evaluation
  - Explain metrics
  - Discuss evaluation results for all models that classify readability
    - Include many, many graphs -> examples in scripts

    - Potentially ask people what they think of the application
        - May require clearance -> in which case, do not
    - Discuss implication of evaluation for model selection
- Chapter 6: Conclusion
    - Limitations
    - Opportunities for Future Work
- Appendix
    - Link to code
    - Sample data
    - Bibliography

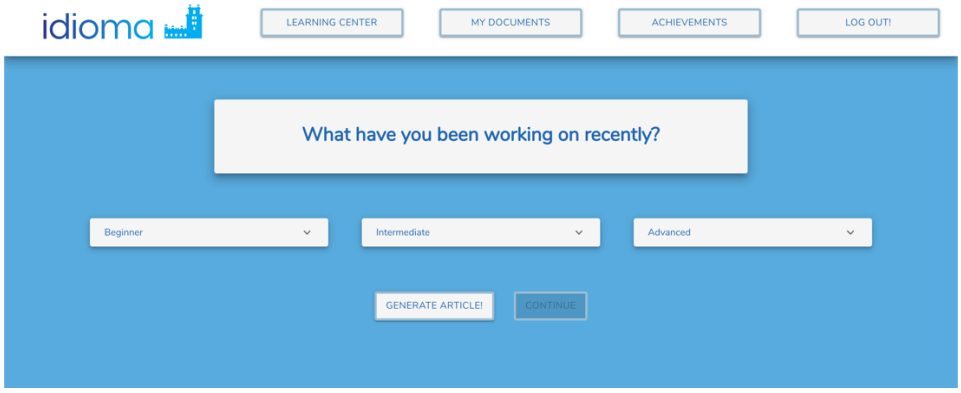
Progress thus far (by component):

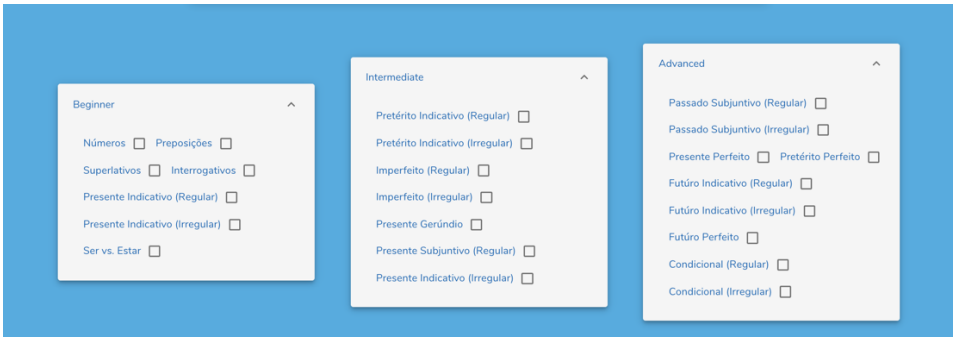
- Writing
    - Draft paper completed
- Authentication
    - Implemented:
        - Users required to log into web app
        - Web app utilizes Auth0 authentication module to verify user credentials against MongoDB Atlas DB
        - User session preserved through the use of cookies built into Auth0
        - User able to log out of web app



    - Limitations / TODO (Priority [1-5]):
        - Protect private paths so that unauthorized users are bounced to the login screen (2)
- Front-end
    - Implemented:
        - Home Page
            - Displays logo, various stylistic elements, and a login button

- Learning Center Page
  - Intended to be the central hub of the web app
  - Asks user for which Portuguese grammatical structures they would like to practice and, after form submission, generates a Document ID, corresponding to an “appropriate” document available on the MongoDB Atlas database
  - Includes a navigation header as well as a button to move to a page displaying the generated document





- Document Page
  - Dynamically-generated page displaying the “appropriate” document generated on the Learning Center page

- Limitations / TODO:
  - Home Page
    - CSS fixes (1)
  - Navigation Header
    - Buttons for navigation to “Achievements” and “My Documents” pages (3)
  - Document Page
    - Module displaying the grammatical structures found in the document, for user reference (5)
    - Dynamically generated color-coding to indicate where each grammatical structure is present within the text (4)
  - Achievements Page
    - Module to display all of the achievements earned by the user (3)
    - Module to display all of the achievements a user has yet to acquire (3)
  - My Documents Page
    - Module to display all of the documents read by the user (3)
    - Filtering system that allows users to sort and filter by grammatical structures within read documents (3)

- Back-end
  - Implemented:
    - API Functions:
      - '/api/getuserExperience: takes in user email; returns list of the number of times the user has seen each feature in a document
        - Straightforward database query
      - **'/api/getArticleId':** takes in user email and list of structures user would like to practice; performs matching algorithm and returns “appropriate” document ID
        - By far the most important API function

        - First filters out (from a list of all available documents) those documents that have been seen by the user, and those documents that do not match their level: “beginner” or “advanced”
        - Constructs a dependency tree over all candidate Portuguese grammatical structures, and then uses the *toposort* library to construct a topological sort over those nodes
        - Samples 100 documents, calculates the relevance and prevalence of each feature within each document, and inputs those values to a formula that outputs an “appropriateness threshold”
            - Formula:
            - $\boldsymbol{TODO - equation}$
        - Randomly samples a single documentfrom the subset of the original 100 documents that meets the appropriateness threshold and returns its ID
    - '/api/updateUser: takes in user email, as well as ID and feature list of article most recently viewed; adds document to list of documents viewed by user, adds document feature list to user feature list, and updates database with user object
        - Another straightforward database query
- Misc. Functions:
    - ‘top_sort_topics’: uses the *toposort* library to create a topological sort over the grammatical structure dependency nodes
    - ‘model_documents’: applies the preferred model to a set of candidate documents, determining whether it is simple (‘s’) or difficult (‘d’)
    - ‘featurize_documents’: applies a POS tagger to each candidate text, traverses said text, and returns a list with the counts of each feature within the document
        - Utilizes a pre-trained Portuguese convolutional neural network provided by spaCy to tag the candidate texts with the relevant part of speech, as well as with other useful information
- Limitations / TODO:
    - The majority of the back-end work should be complete at this point, save for algorithmic updates and functions for CRUD operations on the Atlas database (1)

- Data:
    - Completed:

- Attempted to obtain data from Pedro Curto and Nuno Mamede, co-authors of the article entitled, "Automatic Text Difficulty Classifier: Assisting the Selection of Adequate Reading Materials for European Portuguese Teaching"
    - They pointed me in the direction of the *Instituto Camões*
- Attempted to obtain data from the *Instituto Camões*, a Lisbon-based institution dedicated to the international promotion of the Portuguese language and of Portuguese language-learning methods
    - My contact fell through, and I was unable to secure the data
- Attempted to obtain data from Jorge Alberto Wagner Filho, author of the article entitled, "Automatic Construction of Large Readability Corpora"
    - Successfully secured that data, which I then used in the training and testing processes

o Description:

- Dataset size: 5,325 instances
    - Training set size: 1,278 instances
    - Validation set size: 2,982 instances
    - Test set size: 1,065 instances
- Example instance:

words	syllables	letters	unique	ttr	flesch-adap	coleman	flesch-grade
765	1539	3781		0.4339869281	54.29075066	11.90755556	16.89686017

ari	awl	awlstd	psfl	zh	wiki	brescola
12.77766947	4.94248366	3.164234211	d	d	s	s

    - Each instance originally contained <u>much</u> more information, such as parsing data, parts of speech, and scores that we were unable to compute
        - The above are all of the features that we computed for our larger corpus, and thus the set of features used in all of the modeling processes
- Feature Descriptions:
    - Note: In "Experimental IR Meets Multilinguality, Multimodality, and Interaction," Fabio Crestani et al. provide several adjusted readability formulas for Portuguese. In these cases (marked with an asterisk), both were implemented and used the adjusted metrics.
    - 'words': number of words in text
    - 'syllables': number of syllables in text
    - 'letters': number of letters in text
    - 'unique': number of *unique* words in text
    - 'ttr' (type-token ratio):

        - $\frac{unique\ words}{total\ words}$
    - ‘flesch-adap’ (Flesch reading ease):
        - $206.835 - 84.6 \times \frac{total\ \ syllables}{total\ words} - 1.015 \times \frac{total\ words}{total\ sentences}$
    - ‘fern-huerta’ (Fernandez Huerta readability formula)
        - Note: Not used in modeling and not present in data, but useful for certain comparisons
        - $206.84 - 60 \times \frac{total\ syllables}{total\ words} - 1.02 \times \frac{total\ sentences}{total\ words}$
    - ‘coleman’ (Coleman Liau index)
        - $5.88 \times \frac{total\ letters}{total\ words} - 29.6 \times \frac{total\ sentences}{total\ words} - 15.8$
        - $* 5.73 \times \frac{total\ letters}{total\ words} - 171.365 \times \frac{total\ sentences}{total\ words} - 6.662$
    - ‘flesch-grade’ (Flesch-Kincaid reading level)
        - $0.39 \times \frac{total\ words}{total\ sentences} + 11.8 \times \frac{total\ syllables}{total\ words} - 15.59$
        - $* 0.883 \times \frac{total\ words}{total\ sentences} + 17.347 \times \frac{total\ syllables}{total\ words} - 41.23$
    - ‘ari’ (Automated Readability Index)
        - $4.71 \times \frac{total\ letters}{total\ words} + 0.5 \times \frac{total\ words}{total\ sentences} - 21.43$
        - $* 6.286 \times \frac{total\ letters}{total\ words} + 0.927 \times \frac{total\ words}{total\ sentences} - 26.551$
    - ‘awl’: average letters per word
    - ‘awlstd’: standard deviation of letters per word
    - ‘psfl,’ ‘zh,’ ‘wiki,’ ‘brescola’:
        - In his paper, Filho trained 8 classifiers on 8 unique corpora. These features correspond to the outputs of 4 of these corpora on each instance

- Models
    - Completed:
        - Developed a suite of 8 different models
        - Trained each model on a randomly-sampled subset of the above data (see ‘Data’ section for details)
        - Derived evaluation metrics for each of the models
        - Selected a well-performing model and classified corpus of documents
    - TODO:
        - Ensure validity of testing metrics (4)
        - Perform more rigorous analysis of model performance (4)
        - Visualize results (3)
    - Implemented:
        - ‘master.py’:

        - Trains each model in succession, invoking a script for each one that uses packages from the *scikit-learn* library
    - 8 model scripts -> each one performs hyperparameter optimization, feature selection, training, testing, and evaluation:
        - ‘decision_tree.py’
            - Utilizes a decision tree model
            - Not terribly complex but a great baseline model
        - ‘nearest_neighbor.py’
            - Utilizes a *k*-nearest neighbor model
            - Provides significant flexibility in choice of *k*, the neighbor count, while still being a relatively straightforward model
        - ‘sgd_classifier.py’
            - Utilizes a linear model that implements stochastic gradient descent for learning
            - Also has significant flexibility in the choice of loss function, but the wide variety of hyperparameters proved to be a challenge
        - ‘mlp_classifier.py’
            - Utilizes a multi-layer perceptron model with different loss functions
            - Can learn non-linear relationships, but some implementations necessitate non-convex loss functions, which make learning potentially challenging
        - ‘gaussian_nb.py’
            - Utilizes a straightforward Naïve-Bayes model
            - Very simplistic, but another great baseline
        - ‘bagging.py’
            - Utilizes an ensemble of Decision Stumps
            - Interesting to see if an ensemble method would work better
        - ‘random_forest.py’
            - Similar to ‘bagging.py,’ but uses bootstrap sampling as well
        - ‘svc.py’
            - Utilizes a classifier built on a support vector machine
            - Provides flexibility in the various kinds of possible kernel functions
    - Evaluation for all 8 models:
        - Example:

```
DECISION TREE: acc: 0.92 precisions_by_label [0.91, 0.92] precision_global 0.92 recalls_by_label [0.89, 0.93] recall_global 0.92 f1s_by_label [0.9,
0.93] f1_global 0.92
```

        - Evaluation Metrics:
            - Accuracy
            - Precision (for each of 2 labels / global)
            - Recall (for each of 2 labels / global)
            - F1-Measure (for each of 2 labels / global)
- Database
    - Completed:
        - Successfully deployed a cluster on mongoDB Atlas, a service that hosts user databases on mongo-created cloud networks
        - Created collections for users, achievements, features, and documents, the last one containing all of the documents scraped, featurized, and classified by earlier parts of the pipeline
        - Fully integrated database with back-end; changes prompted on the front-end correctly pull from and update the corresponding collection on the Atlas cluster
- Web Scraper
    - Completed:
        - Web scraper that uses the scraping library *Scrapy* to pull articles from several categories available on *Diario Notícias*, a prominent Portuguese web newspaper
        - Collected 1,815 articles from the above website, for use later on in the pipeline
    - TODO:
        - Build scrapers for other Portuguese news sites (2)
- Web Hosting
    - TODO:
        - Full web hosting stack (2)