Idioma: A Document-Based Language-Learning Platform

Paulo Frazão

Adviser: Professor Xiaoyan Li

Submitted in partial fulfillment

of the requirements for the degree of

Bachelor of Arts

Department of Computer Science

Princeton University

May 2020

I pledge my honor that this thesis represents my own work in accordance with Princeton University regulations.

/s/ Paulo Frazão

Abstract

Despite ever-increasing demand for language-learning tools and, specifically in a second-language context, there is a concerning lack of reading comprehension tools in the market. Modern applications, such as Memrise and Duolingo, provide excellent systems for students undertaking a foreign language; however, the majority of these tools fail to provide a learning experience that focuses on the development of reading comprehension skills while also maintaining an acceptable level of interactivity and personalization. The goal of this project is to implement a solution that addresses these concerns using cutting-edge machine learning and natural language processing techniques. Idioma is a Portuguese language-learning web-app that presents the user with articles customized to their proficiency level and interests. The application uses a series of machine learning classifiers to label web-scraped content, and then employs a proprietary selection algorithm to offer the user content that is both engaging and appropriate to their skills. The application grows with the user, tracking the content that they consume to dynamically refine the selection algorithm, ensuring consistent up-to-date performance. Furthermore, the application boasts a variety of quality-of-life and gamification features with the intention of maximizing user entertainment and retention. Idioma is implemented using a combination of ReactJS, Flask, MongoDB, and Scrapy. The models underlying the application were trained and evaluated using scikit-learn libraries; the best-performing models achieved an accuracy of 80% and comparable precision and recall, demonstrating a fair competency in this binary classification task. There exist a number of opportunities for future extension in this project, but it nonetheless offers a significant foundation towards the task of building a dedicated, exciting, and cutting-edge reading comprehension tool suite.

Acknowledgements

I would firstly like to thank my advisor, Xiaoyan Li, for her time, patience, and incredible insight on this project. From my first independent work seminar with her last spring, I knew that she would challenge me to push my ideas as far as they could go, and she did not disappoint.

I would also like to thank Nicola Cooney, my second reader. Throughout my time as an undergrad, she has helped me to cultivate my love for the Portuguese language, and it is to her that I owe the passion that drove me to turn my idea into a reality.

I also owe great thanks to Jorge A. Wagner Filho, who contributed the data set upon which my entire application rests. He selflessly offered his work, and quite a bit of time to re-purpose it for the project's needs, and for that I am incredibly grateful.

Lastly, I would like to express thanks for all of my family and friends, who are too many to name. You all give me the courage and the passion to pursue the things that I love, and I cannot express how grateful I am to have your love and support.

To my parents, who instilled in me a love for the language and culture of my ancestors. *Vocês são o meu mundo.*

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Problem Background

As the world continues to chug towards globalization and approach an ever-growing state of multicultural awareness, one might find it increasingly difficult to argue against the merits of learning a foreign language. For individuals of all backgrounds and ethnic identities, languages provide a bridge for mutual understanding, particularly with those with whom few other experiences are shared. And key to this process of language learning is the skill of reading comprehension. In their article, *The Literacy Dictionary: The Vocabulary of Reading and Writing*, Harris and Hodges define reading comprehension as "the construction of the meaning of a written or spoken communication through a reciprocal, holistic interchange of ideas between the interpreter and the message" in a way that "is influenced by that person's prior knowledge and experience" [14].

At first, one might feel inclined to doubt the importance of developing strong foundations for reading comprehension, specifically in the context of second-language learning. A number of studies have provided evidence to the contrary, however. Hagtvet, for one, performed a series of tests on randomly-selected nine-year-old chil-

dren to examine the potential for interdependence among the growth of their skills in reading comprehension, listening comprehension, and decoding (the process of applying knowledge of letter-sound relationships to quickly recognize and pronounce unfamiliar words) [13]. Figure 1.1 captures the results of this study, displaying the correlations between various language measures of the three categories above. It is notable that measures of reading comprehension (i.e. oral story retelling, written story retelling, oral cloze, etc.) consistently correlate highly with all other implemented features; therefore, we may presume that, at least among those surveyed and perhaps in the more general language-learning population, strength in reading comprehension may engender strong listening comprehension, decoding, and, ultimately, competence in a first- or second-language.

	1	2	3	4	5	6	7	8	9	10
1 Phonemic awareness	1	0.57**	0.39**	0.43**	0.44**	0.39**	0.47**	0.66**	0.52**	0.63**
2 Vocabulary		1	0.49**	0.39**	0.49**	0.46**	0.53**	0.50**	0.39**	0.75**
3 Syntax			1	0.39**	0.39**	0.42**	0.41**	0.50**	0.43**	0.52**
4 Digit span				1	0.41**	0.36**	0.45**	0.49**	0.40**	0.54**
5 Decoding					1	0.42**	0.48**	0.50**	0.44**	0.41**
6 Oral story retelling						1	0.65**	0.54**	0.49**	0.44**
7 Written story retelling							1	0.56**	0.39**	0.46**
8 Oral cloze								1	0.64**	0.56**
9 Written cloze									1	0.44**
10 IQ										1

**$P < 0.01$.

Figure 1.1: Correlations between language measures of reading comprehension, listening comprehension, and decoding [13].

Despite its apparent importance, however, it is also the case that reading comprehension remains a challenging skill to acquire and master, both in the cases of first- and second-language learning. This has become especially relevant in the United States, which has shown a steady decline in the reading comprehension abilities of those in its public school system. According to the National Assessment of Educa-

tional Progress, an exam administrated to Grades 4 and 8 every few years to track state-wide and national learning trends, many more states suffered losses in generalized reading ability than incurred gains; only one state demonstrated improvement in each of Grades 4 and 8, while 17 states experienced a significant score decrease in Grade 4 classes and 31 experienced the same effect in Grade 8 [6]. Furthermore, this issue becomes more challenging when we consider reading comprehension in the context of second-language learning. Droop and Verhoeven conducted a series of studies in which they attempted to gauge the relative ease of developing reading comprehension in a target language among students with an existing knowledge of that language and students who are being exposed to it for the first time. Through their experimentation, they provided methodological backing to the perhaps-obvious conclusion that "the differences in reading comprehension scores between first and second language learners [the former being significantly higher than the latter] are due to differences in knowledge of the target language" [17]. It is clear, then, that reading comprehension is a serious undertaking when practiced in the context of a language already known, and this challenge increases considerably when it is extended to the realm of second-language learning.

It is clear, then, that there exists demand for some kind of system that would empower individuals (in particular, those seeking to learn a new language) to develop stronger reading comprehension skills and, therefore, more quickly progress in their foreign-language competence. But the question remains, however, of how best to frame the task of reading comprehension such that learners feel an intrinsic motivation to develop these skills on their own. In her work, "Reading Comprehension Strategies: Theories, Interventions, and Technologies," McNamara explores the various drivers that push a student to develop their reading abilities; Figure 1.2 illustrates the web of dependencies that, according to the pedagogical theory adopted by Hagtvet, beget achievement and knowledge in the realm of reading comprehension.

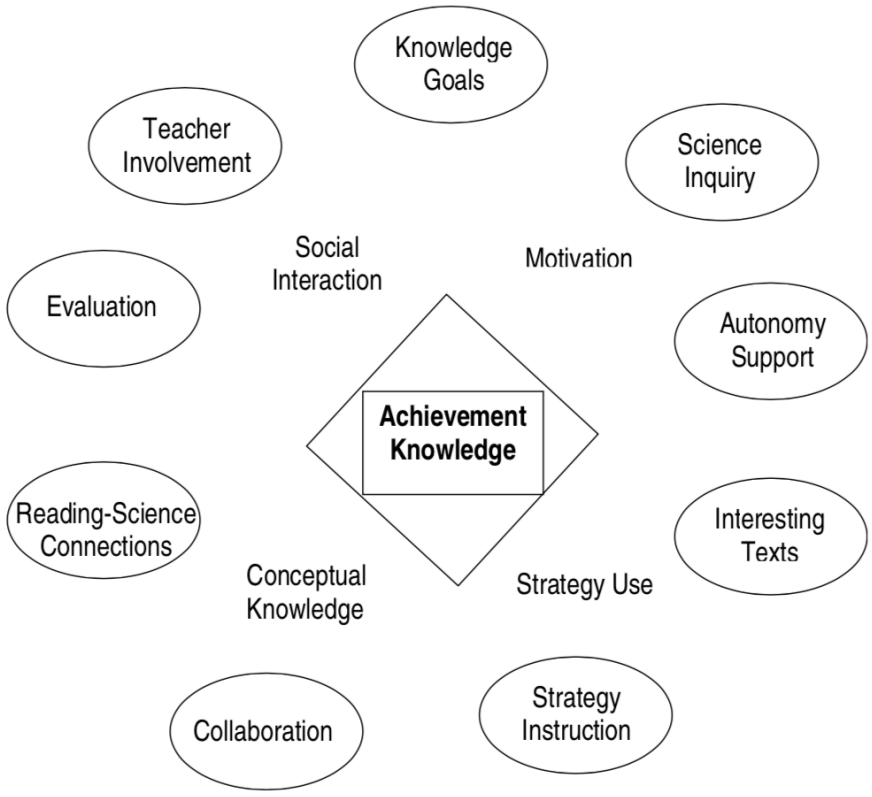

Figure 1.2: The engagement model of reading development used by McNamara [16].

In a manner similar to the other aforementioned researchers, McNamara applies quantitative methods in an attempt to concretize the relationships embedded in this diagram. By surveying a group of Grade 4 students, she successfully demonstrates that intrinsic motivation is key for students looking to develop knowledge of a second language, and that "motivation should be included explicitly in instruction that attempts to increase reading comprehension," a conclusion that was also corroborated by researcher Sarah Logan in her own article [16] [15]. Both individuals leave open the question of how best to provide this motivation, choosing only to elucidate its importance in this particular learning process.

1.2 Motivation and Goal

The above being noted, then, the motivation for this project becomes clear. Not only does there exist demand for a system that will improve learners' reading comprehension and, by extension, their overall language ability, but this demand also presents an excellent opportunity to ground such a system in techniques of motivation-driven learning. An ideal system would be one that supported a user's goals in the service of reading comprehension, but that allowed for them to craft those goals themselves and to chart their own course through them, providing agency and other reinforcers of the intrinsic motivation and McNamara and Logan esteem.

This project aims to provide exactly such a system for student use. The goal of my research was to utilize state-of-the-art natural language processing techniques to create a full-stack web-application that would provide users with a service that a) allows them to develop their reading comprehension while b) exercising agency and experiencing greater motivation as a result, in the hopes of making the learning process more attractive than existing solutions. While this was admittedly a very large undertaking, and the current implementation is tailored exclusively to individuals seeking to learn Portuguese, I believe that this project has made significant strides towards achieving both of these objectives.

Chapter 2

Related Work

Before introducing my own approach towards this problem, it is worth spending some time considering the ways in which other individuals and groups have tackled the problem of reading comprehension, specifically through a lens of machine learning and motivation-building.

This chapter will introduce the reader to some existing solutions at varying levels of granularity. It will begin by considering competing full-stack applications, such as Memrise and Duolingo, that focus not on reading comprehension, but on the language-learning process as a whole. It will then narrow its focus, centering on those systems that enable users to hone their reading comprehension abilities in a language of their choosing. Given that natural language processing is a substantial component of my own approach to this problem, this section will finish with a discussion of NLP and current work integrating it into comprehension-adjacent fields.

2.1 Language-Learning Applications

Let us consider, first, those applications that are already in the hands of students around the world. Thankfully for prospective language-learners, this field is a popular one; there exist dozens of marketed systems that are geared towards helping the

individual acquire competency in a second language. This section will cover only a couple of these applications, highlighting both their strengths and the areas in which improvements might be made.

One such system is Memrise, a platform developed and eventually published by Greg Detre, Ed Cooke, and Ben Whately in 2005 [5]. Unlike many of its competitors, which offer highly-refined experiences often crafted by professionals in the field, Memrise relies on user-generated content; users can publish "Mems" (pictorial mnemonic devices aimed at reinforcing learning) or entire courses, such as the "Beginners Portuguese" curriculum published by user "worrybugger" and pictured below.

Figure 2.1: A description for the "Beginners Portuguese" course on Memrise, published by user "worrybugger."

In many ways, Memrise is the ideal language-learning application. It offers, for one, the basic motivational accoutrements that most have come to expect from these kinds of services: achievements, leaderboards, and other details that compel the user to feel rewarded and, thus, to keep coming back. Furthermore, the separation of lessons into courses that have been crafted by other "Memrisers" enables the user to create their own language-learning experience; with lessons on emotions [4], animals [8], and even business vocabulary [2], there is a level of potential for customization here that few other applications have achieved.

That being noted, however, the service does have a number of limitations, the most important being the fact that it does not offer very much in the way of reading comprehension; in fact, it only allows users to learn and practice vocabulary and

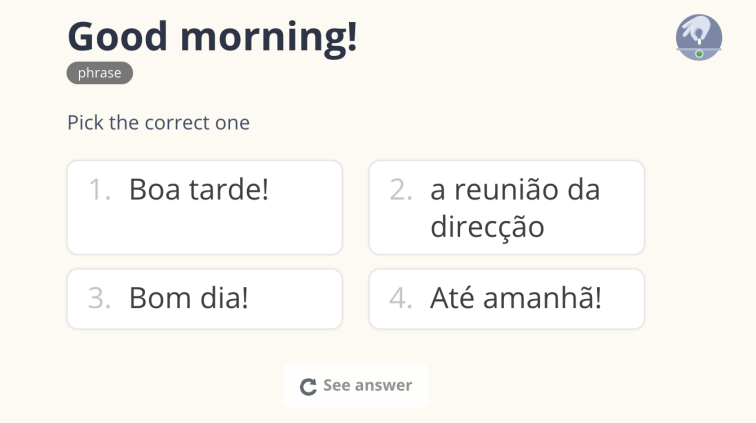

Figure 2.2: A sample question presented to a user of Memrise.

short phrases, despite the personalization. Figures 2.2 and 2.3 show a sample quiz question and results screen displayed to users, respectively. As is clear, the questions are scarcely more complex than those offered by flashcard services like Quizlet. The results, consequently, are centered exclusively on the individual's ability to recall this vocabulary, and so do not provide a very good understanding of how thoroughly the user actually understands the language. This service is popular, and rightfully so, but it does not offer any opportunities for the user to develop their reading comprehension.

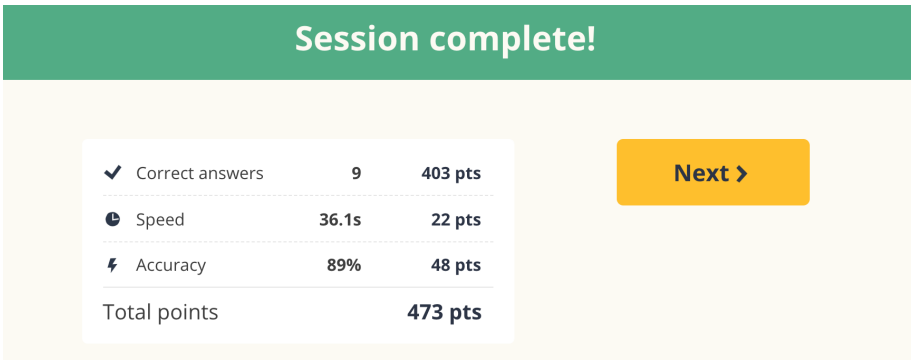

Figure 2.3: The results screen displayed to a Memrise user upon completion of a lesson.

Duolingo, another even more widely-used application, suffers from similar issues. Launched publicly in 2012 by Luis von Ahn and Severin Hacker, this platform also offers users the opportunity to immerse themselves in curricula covering over 30 languages [9]. Like Memrise, it too boasts many fun, quality-of-life features geared at

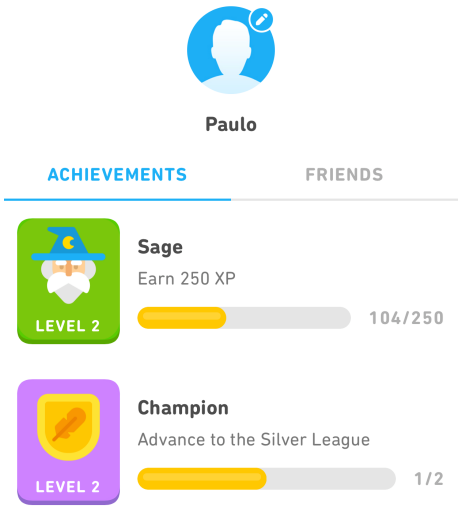

Figure 2.4: The achievements screen in the Duolingo mobile application.

maximizing user retention; I found its achievement system, pictured in Figure 2.5, to be particularly successful in its style.

Naturally, it stands out from competition like Memrise in a number of key ways. Instead of relying on user-submitted content, for one, Duolingo instead employs experts to hand-craft their courses. As a result, their offerings feel much more polished then those observed above, and they are more comprehensive as well. Duolingo does not draw the line at vocabulary; their curricula interweave modules on vocabulary and all (or most) important grammatical and syntactical structures. This technique is visible in Figure 2.5, which depicts the introductory sequence for the Duolingo course on German. This is a significant improvement. Vocabulary is important, but arguably useless without the grammatical lessons necessary for reading comprehension, listening comprehension, and attempts at discourse.

That said, however, Duolingo does leave something to be desired, particularly in the way of reading comprehension. Its lessons are thorough, and it is possible that they offer some reading practice; admittedly, my progress in German has not been consistent enough to unlock their "Stories" section. Nevertheless, my research has brought me to the conclusion that reading comprehension is not Duolingo's focus, and

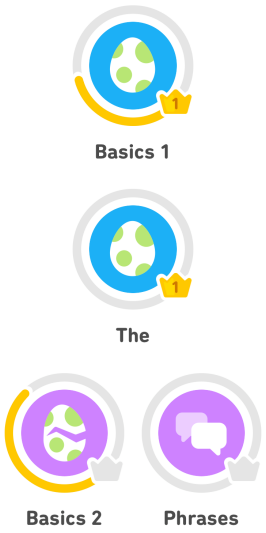

Figure 2.5: The progression screen for the German language curriculum in the Duolingo mobile application.

they do not pretend as such. But this does allow the vacuum in dedicated reading comprehension tools to persist, a need that my own implementation aims to fulfill.

2.2 Reading Comprehension Systems

Despite the lack of reading comprehension systems in the above applications, there are a number of smaller-scale dedicated tools available on the market. Let us now consider a handful of these, and how my own implementation aimed to correct their shortcomings.

The most straightforward option that I found was the service provided by "lingua.com," a platform that offers "the tools and resources you need to make learning a language easy and fun" [19]. This website provides users with carefully curated texts and quizzes geared towards helping students of different proficiency levels practice their reading. Interestingly, Lingua actually bases its metrics of proficiency off of the CEFR (Common European Framework of Reference for Language), a universal standard developed in Europe in 2001 that offers a language-agnostic description of an

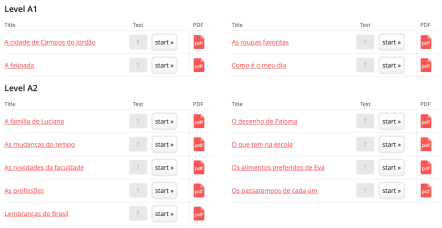

Figure 2.6: A subset of the texts offered in the "Portuguese Texts for Beginners" course on Lingua.

individual's linguistic competency [3]. Nevertheless, Lingua offers a fair assortment of texts for each level, covering an equally impressive array of subjects, as pictured in Figure 2.6.

While functional, this approach is not terribly thorough in its potential for personalization nor in its dynamism. On the one hand, users may practice on texts corresponding to their present level of proficiency, but they have minimal control over the subject matter of these articles or over the specific linguistic features that they are being shown. On the other, the quizzes provided limited feedback to the user; it is hard enough to ascertain how one is growing after each text, much less to understand what still needs more reinforcement. As such, Lingua is an excellent beginner's tool but it does fall short in these respects.

Readlang, a service developed by Steve Ridout, a senior software engineer at Duolingo, exists at the other end of the cuztomizability spectrum [18]. This application integrates directly with most web browsers, allowing a user to dynamically translate any words that they come across and turn them into flashcards for later viewing. In this way, it offers a user complete control over the subject matter on which they practice, as well as the potential for greater influence over the specific grammatical structures that they see (in the cases of very specific reading materials). Unfortunately, it lacks many of the features contained in applications explored above, including motivation-building gamification techniques (i.e. achievements, leaderboards) and any real feedback to guide the student in their exploration of the language. Read-

lang is an excellent supplementary language-learning service, but it does not have the functionality to carry the full weight of a second-language curriculum.

It is clear, then, that even those applications geared towards reading comprehension leave something to be desired. While some offer the same quality-of-life features that can be observed in more sweeping services such as Memrise and Duolingo, others opt to disregard those systems in favor of user customization. There is a need for a service that combines these two poles, and the approach presented in this paper will do just that.

2.3 Machine Learning and Readability

Before moving on, however, this section will consider existing work in one final field of great relevance to my own implementation: natural language processing, particularly in the context of text readability. The specifics of its application to this project are presented in the Approach and Implementation sections below, but it is worthwhile to consider the inroads that others have made within this area before describing my own paper's contribution.

In terms of recent work in the field of Portuguese language readability classifiers, Pedro Curto et al.'s 2014 article, "Automatic Text Difficulty Classifier: Assisting the Selection Of Adequate Reading Materials For European Portuguese Teaching," was a crucial starting point for my research [11]. They developed a system that uses NLP methods to extract a variety of linguistic features from a candidate text and that then uses a series of pre-trained classifiers to label that document according to the CEFR categories discussed above. Their best-performing algorithm achieved an accuracy of 75.11% for this five-level classification problem [11], and the article invites the reader to demo this system at the following website: https://string.hlt.inesc-

id.pt/demo/classification.pl. [1] Nevertheless, this article provided an excellent starting point for my research in these classifiers, and it pointed me in the direction of another crucial source.

The source in question was Jorge Alberto Wagner Filho et al.'s 2016 paper entitled "Automatic Construction of Large Readability Corpora" [12]. Filho et al. applied a similar kind of methodology as Curto and his colleagues; they developed a system that relies on pre-trained machine learning classifiers to sort candidate texts based on their readability. Where they differ, however, is in how they use these classifications. Filho et al. implemented this system in the hopes of integrating it into a larger pipeline that would crawl the web and automatically categorize the Brazilian Portuguese web pages that it found and enumerate them in a global repository. This database became known as the BrWaC (Brazilian Portuguese Web as Corpus). Containing over 3.53 million documents and 2.68 billion tokens, this repository is the largest accumulation of annotated Brazilian texts available today [1]. Filho and his project would prove instrumental in supplying me with the data that I needed to develop my own implementation, a subject to which this paper will return.

[1] Unfortunately, at the time of writing this service appears to be offline. Requests for "Classification" and "Feature Extraction" return only a server error.

Chapter 3

Approach

In order to address the opportunities for growth that were noted in the section prior, I decided to develop my own full-stack web application named Idioma. This section will provide a brief overview of my approach as well as outline a few reasons why this strategy corrects a number of the issues noted in other, existing language-learning applications.

3.1 Overview

In essence, Idioma is a web application that puts the control in the hands of the reader; upon logging in, a reader can search for reading material based on its subject matter, the grammatical or syntactical structures that it contains, or simply the text's general difficulty level. The system will then use a combination of machine learning classifiers and a homegrown selection algorithm to present the reader with the most appropriate document given their selection criteria. All the while, the system will dynamically track the user's progress, logging those changes in their profile to continue providing appropriate documents during future selection periods. Idioma aims to merge the deep customizability of packages like Readlang with the quality-of-life features and comprehensiveness of more fully-fledged applications such as Memrise or Duolingo.

Although it only supports Portuguese in its current state, this system should be easily extensible, both to support new languages and new document sets.

3.2 Approach Advantages

Now, the next section will provide a closer look at my implementation of this system. It is worth noting, first, the benefits that such a system would provide, especially in comparison to existing competition on the market:

1. **Ease and Simplicity**

    (a) Despite being a great deal more elaborate than many of its competitors, Idioma (as it is described above) creates an exceptionally simple experience for the user. From their prospective, all that is asked of the user is to log on, select some criteria of interest, and read. Of course, there is quite a bit happening on the back-end in order to provide them with the best possible match and to keep track of their profile; by encapsulating this away from the student, however, they are left with an interface that is more user-friendly than much of the competition.

2. **Consistency of Experience**

    (a) One pitfall of services that aim to suggest content for a user (i.e. StumbleUpon, Reddit, etc.) is that they might provide inconsistent experiences for the individual over time. While one suggestion might be particularly enjoyable, there is a chance that the next one will not be, a hiccup that may push the user away from the platform. By applying the same algorithm to a dynamically-changing user profile for each document requested, however, Idioma optimizes for consistency. By tuning a few algorithmic parameters, the developer can control just how quickly the user is assumed

to learn, providing as gradual a growth experience as is deemed necessary for the student.

3. **User Retention**

   (a) This is the clearest benefit of the approach described above and, consequently, the aspect that my implementation works the hardest to ensure. My goal for Idioma was, primarily, to create an experience that users would enjoy and, consequently, want to undergo again. To that end, I ensured that this approach would be flexible enough to insert as many quality-of-life and gamification mechanics as I would have time to develop. By keeping track of user progress, the door is opened to achievement systems, leaderboards, and countless other features that will encourage students to keep returning. Furthermore, the focus on user agency, as exemplified by their ability to specify the thematic and grammatical content, also encourages the student to keep on reading, practicing the skills that they need to supplement their understanding of the language.

Clearly, this approach offers a number of benefits over existing solutions, and I would argue that this is due to the fact that it takes the best from all of its competition. By merging the form factor and comprehensive experience offered by Duolingo and other leading language-learning platforms with a complete focus on reading comprehension, an oft overlooked task that is supported only by smaller projects like Lingua or Readlang, it aims to create the fully developed reading experience that students of a second language deserve.

Chapter 4

Implementation

With the goal of demonstrating how Idioma addresses the concerns listed in the section above, this chapter will provide a thorough explanation of my implementation of this language-learning tool. It will begin with a brief overview of the system architecture, with equally brief descriptions of each component present within it. It will then consider each of these components in turn, explaining their inner workings in greater detail. It will conclude with a discussion of potential use cases, both from a technical and from a practical standpoint.

4.1 System Overview

Given that I chose to develop Idioma as a full-fledged web application (rather than as a set of scripts or CLI tool, for instance), its architecture is quite involved. This section will provide an overview of this construction, which is outlined in ***Figure 4.1***.

The core purpose of Idioma is to provide the user with carefully-curated documents, selected using a combination of machine learning classifiers and my own algorithm. The major components involved in the retrieval and display of these documents are the front-end and back-end; the former takes requests from the user and

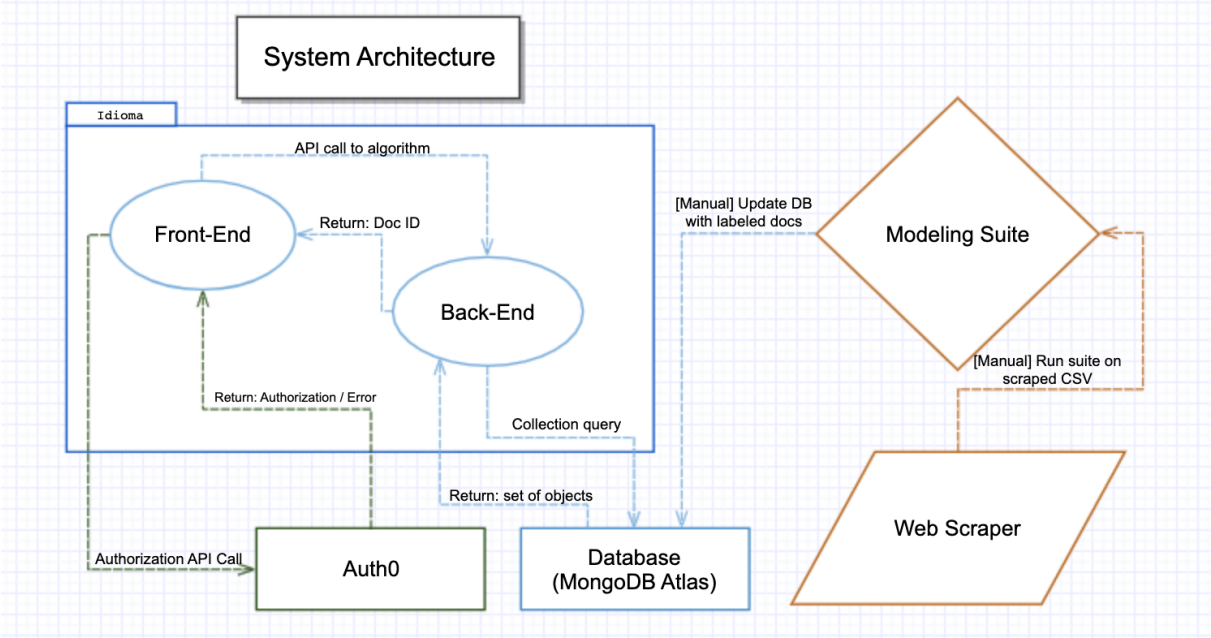

Figure 4.1: An architecture diagram for Idioma, developed using Gliffy.

displays the responses to said requests, while the latter is responsible for the construction of these responses. By the time that the user requests a document, however, the majority of the work has already been done; the web scraper and modeling suite execute asynchronously from the rest of the application. Whenever the developer of the application would like to update the set of documents that Idioma furnishes to users, they must launch the web scraper anew. Upon execution, these scripts build a database of Portuguese news articles, outputting the result in a CSV located in the project directory. To classify these documents, the modelling suite must be executed; it retrieves that file, performs its classifications, and outputs the results in a final CSV. All that is left is for the developer to add this file into the corresponding MongoDB Atlas cluster for the back-end to access in real time.

The rest of the system comes into play as the user logs in and uses the application. Before accessing their account, a user must log in through Auth0, an authorization platform that interfaces well with Flask and with MongoDB Atlas. Upon validation, the user can access the front-end through a secure session; they may request articles,

continue their learning, or peruse their achievements and reading history. Each of these functionalities necessitates its own back-end implementation, which will be discussed later in this chapter. Nevertheless, the user session terminates when they log out through Auth0, bringing their learning to a temporary close.

4.2 Component Implementations

4.2.1 Data

In my development of Idioma, I needed two kinds of data. On the one hand, I needed training data for my classifier suite. In order for my models to be trained, I required a set of Portuguese-language articles that were labeled by difficulty level; this proved to be quite a tough ask. On the other hand, I also needed a much larger set of unlabeled articles that I could classify and then offer to users. This subsection will explain how I secured both collections.

Training Data

I began my search for training data by combing through all of the papers that I could find on readability classifiers and related work for the Portuguese language. This process eventually led me to an article titled "Automatic Construction of Large Readability Corpora," by Jorge Alberto Wagner Filho et al. This article details the construction of a framework for the automatic construction of Portuguese corpora for readability problems; according to Filho and his co-writers, this research led to the development of the Brazilian Portuguese Web as Corpus, an online Portuguese-language database that includes 3.53 million documents, as well as their corresponding textual attributes [12]. Though I was able to get access to this repository, it lacked any kind of readability or difficulty labels, forcing me to try reaching out to the authors

of teh article instead.[1] It was at this time that I entered into contact with Jorge Filho himself, who was immensely helpful in supplying me with serviceable training data.

Filho granted me access to a dataset of 5,325 labeled Portuguese articles. Each instance associated a "docid" (presumably for use in searching for the document in the BRWaC) with a set of over 100 distinct textual features and with the outputs of four classifiers. According to their article, Filho et al. acquired several different labeled corpora, each of them possessing a unique labeling scheme. Therefore, they decided to develop a classifier using each corpus, and then proceeded to compare the results among the models. The dataset provided by Filho contained the outputs generated from models trained from four corpora: *Para o Seu Filho Ler*, a collection of articles for children and for adults, *Zero Hora*, a collection of articles primarily for adults, *Wikibooks*, a set of virtual texts for readers of different proficiency levels, and *Brasil Escola*, a sweeping corpus of educational material for adolescents and teenagers. I was able to use these classifications as a ground truth of sorts in the development of my own classifiers.

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

Figure 4.2: An example instance from the dataset provided by Filho.

Now, as was noted above the dataset provided by Filho was quite extensive. As a result, I decided to trim the majority of features that would be infeasible to compute for a given text (the majority of these features were syntactical in nature; I was not able to procure a library that could break a Portuguese sentence into its constituent units, for instance). ***Figure 4.2*** shows the set of features that were actually used

[1] Access to the BRWaC repository can be requested here: https://www.inf.ufrgs.br/pln/wiki/index.php?title=BrWaC.

in my classification process, and ***Table 4.1*** displays each of these features with its corresponding description.[2]

Feature	Description
'words'	word count
'syllables'	syllable count
'letters'	letter count
'unique'	*unique* word count
'ttr' (token-type ratio)	$\frac{'unique'}{'words'}$
'flesch-adap' (Flesch reading ease)	$206.8 - 84.6 * \frac{'syllables'}{'words'} - 1.0 * \frac{'words'}{sentences}$
'coleman' (Coleman Liau index)*	$-6.7 + 5.7 * \frac{'letters'}{'words'} - 171.4 * \frac{sentences}{'words'}$
'flesch-grade' (Flesch-Kincaid reading level)*	$-41.2 + 0.9 * \frac{'words'}{sentences} + 17.3 * \frac{'syllables'}{'words'}$
'ari' (Automated Readability Index)*	$-26.6 + 6.3 * \frac{'letters'}{'words'} + 0.9 * \frac{'words'}{sentences}$
'awl'	average letters in word
'awlstd'	standard deviation of letters in word
'psfl'; 'zh'; 'wiki'; 'brescola'	classifier outputs

Table 4.1: The features used in classification and their corresponding descriptions.

As is clear, the features used in the classification process were a combination of textual attributes and common readability metrics. We note that those features that are marked with an asterisk in the table above were actually adjusted according to Fabio Crestani et al.'s "Experimental IR Meets Multilinguality, Multimodality, and Interaction." In their book, they examine a set of common readability indices and adjust them for more accurate use when measuring the difficulty of Spanish or Portuguese texts; as such, this project applies their equations where appropriate [10]. Nevertheless, this feature set proved to be quite sufficient for the purposes of classifying the documents that we later scraped, as will be discussed below.

Unlabeled Articles for User Consumption

Given that Idioma aims to provide users with a customizable, interesting learning experience, this project also required a collection of reliable articles covering a wide swath of potential subjects and disciplines. To generate such a collection, I used

[2]The coefficients shown in the table were rounded to first decimal place for presentation.

Scrapy, an open-source framework for crawling websites and extracting data. I settled on this library both because of its extensive documentation and support and because of its seamless Python integration.

Using Scrapy, I was able to develop a full-fledged web scraper that gathered a viable set of articles within 10-20 minutes. This scraper, activated by a simple command line command, traverses the web using a custom-built "spider": a script that specifies the DNS entry points for the scraper, as well as the sequence of actions that the scraper should take on each page. My particular scraper extracts articles from "*Diário de Notícias*," a popular Portuguese newspaper with a particularly straightforward web interface. Upon navigating to its homepage (https://dn.pt/), the spider launches a Selenium Request to dynamically navigate to the homepage for each category of news listed on the site. Once there, it executes a JS script to load as many article thumbnails onto the page as possible, and then follows and extracts the information from each one. [3]

Using this method, the web scraper gathered a total of 1,815 unique articles for use in the web application. While a finalized version of the project would use content from several different sources, the articles from *Diário de Notícias* sufficed for this stage of development.

4.2.2 Models

Scripts

Perhaps the most important individual component of the project, the modeling suite that makes up the core of the selection process is made up of eight distinct classifiers, each of which attempts to determine whether a document is "simple" or "difficult."

[3] At the time of writing, this process was a bit unstable. For reasons unknown, the web scraper would often fail after executing the JavaScript code, especially if pressed to load article thumbnails too many times. Thankfully, the scraper was nonetheless able to generate the requisite articles as needed.

This section will step through the modeling scripts, examining how each model is constructed before offering an explanation of each one and how it fits into the Idioma user flow. [4]

The modeling suite is actually composed of only two Python scripts: a master script, which initiates the construction process and visualizes results, and a more dynamic utilities script, which performs the training, hyper-parameter tuning, and eventual evaluation. The scripts make use of scikit-learn, a very popular machine learning library for Python; it offers a range of easily-implementable models, ranging in complexity from decision trees to multi-layer neural networks. Using these packages, the two scripts create the set of eight models from scratch, saving them for use in the application.

When the master script is launched, the program launches an iteration of the utilities script for each desired model type, passing that type as an argument. The utilities script is designed to be dynamic; based on the model that it is tasked with creating (based on the argument passed to it), it adds the necessary variations to the standard training and evaluation processes. The script begins by attempting to establish a baseline off of which to train the model. It is worth noting, before considering the details of this process, that the training data provided by Filho is not technically a ground truth for the problem that my project aims to tackle. Ideally, the classifiers would be trained on a data set that had been hand-labeled by qualified individuals proficient in the Portuguese language. These labels, however, have themselves been procedurally generated by Filho's classifiers; therefore, a classifier that perfectly models the training data would merely emulate Filho's own project. Nevertheless, we were grateful for this data and hoped that it would yet provide a workable pseudo-ground truth, appropriate for the problem.

[4] We note that, after being scraped, the articles also needed to be properly featurized before being of use to the machine learning models. For this purpose, a Python script was developed that automatically calculated all of the features listed in the "Training Data" subsection above for a candidate document. We omit further details for reasons of convenience.

That said, the first module of the utility script attempts to establish a more fine-grain ground truth from the four classifier outputs in the training data. We considered a number of potential strategies. For one, it seemed sensible to just take as ground truth the column that resulted in the highest accuracy for a given model type, in this case the PSFL column. According to Filho, however, this classifier was trained on a corpus whose purpose was "comparing articles for children with articles for adults" [12]. Therefore, we would expect for it to achieve a high accuracy, as the documents should be easily separable into two difficulty classes. We also considered taking the simply majority label over the four outputs, tiebreaking in favor of either class. Upon closer inspection, however, we noted that the *Brasil Escola* corpus is allegedly intended solely for "children and teenagers" [12]. As such, we decided to instead take the simple majority over the other three outputs; this strategy had the added benefits of providing higher accuracy and removing the need to tiebreak in favor of either class.

The script then proceeds to partition the data into training, tuning, and test sets; 64%, 16%, 20% of the instances are allotted to each, respectively. The tuning set is used first for hyper-parameter selection through exhaustive grid search (the specifics of which are discussed for each model type below) and to determine the percentage of features to base the final model upon. The second process is actually done with input from the user; for each graph, the script plots the percentage of features against the resultant metrics as calculated on the tuning set and then asks the user to select an appropriate threshold for training. These plots are pictured in Figures ***4.3*** and ***4.4***.

After the user selects the threshold, the script proceeds to use only that highest-performing proportion of features in the training process; it also writes the features used by each model type to a txt file for use in later featurization. Using standard scikit-learn packages, the script trains and evaluates each model in turn, saving it and

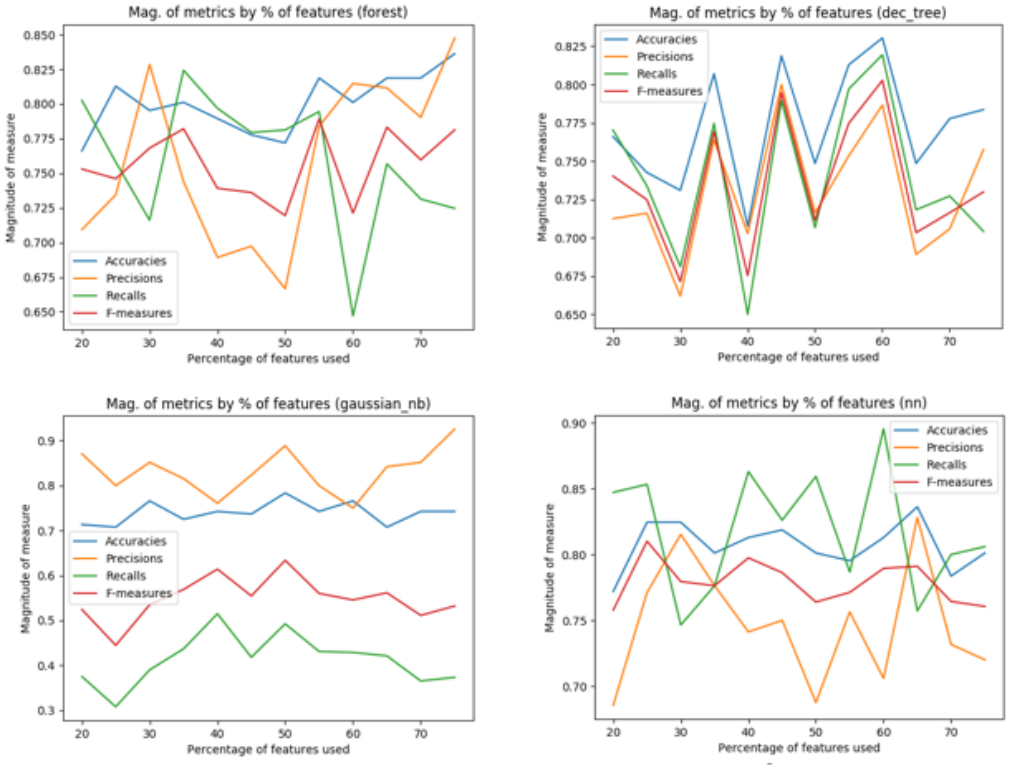

Figure 4.3: Plots of various metrics by feature threshold for Random Forest, Decision Tree, Naive Bayes, and k-Nearest Neighbor models.

its corresponding metrics for the master script to visualize near the end of execution. These results are pictured in the "Results and Integration" subsection below.

Implemented Models

Before considering the results of these experiments, this section will provide a brief explanation of each model in the classification suite. The list of models is as follows:

1. Decision Tree Classifier ("tree.DecisionTreeClassifier()" in scikit-learn)

    (a) Decision trees are arguably one of the simplest forms of classifiers and regressors still used in machine learning today. These models eschew tunable parameters, instead attempting to predict the target value using a simple set of decision rules on the independent variables. While potentially dangerous, as they can easily overfit by making over-complex trees

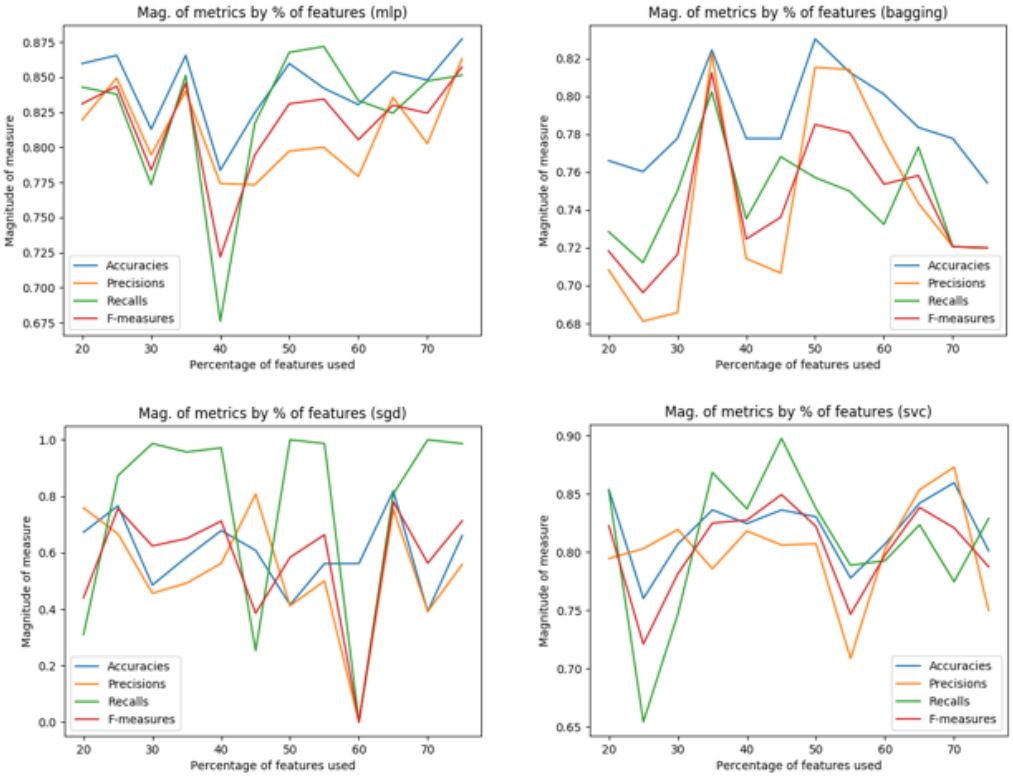

Figure 4.4: Plots of various metrics by feature threshold for Multi-Layer Perceptron, Bagging, Stochastic Gradient Descent, and Support Vector Classifier models.

or simply prove unstable over multiple iterations, they provide an excellent, computation-light baseline against which to measure other model performances.

2. Random Forest Classifier ("RandomForestClassifier()" in scikit-learn)

   (a) The Random Forest Classifier is essentially a way to overcome the shortcomings of the Decision Tree Classifier through model redundancy and randomness. On the one hand, the model actually trains an ensemble of decision trees; when given an input to classify, it simply performs some kind of aggregation over the results of passing that input as argument to each tree, outputting the result. On the other hand, each of these individual trees is instilled with randomness, created from a bootstrapped sample

of instances and trained using a random subset of features. These measures help to prevent the classifier from overfitting on the data, and also (ideally) suppress the effects of outliers on the classification process.

3. Gaussian Naive-Bayes Classifier ("GaussianNB()" in scikit-learn)

   (a) The Gaussian Naive-Bayes Classifier is part of a larger class of models that assumes conditional independence among each pair of features given the value of the label being predicted. This allows the feature to reduce Bayes' formula to the following expression, where y is the target label and x_i is the value of the i-th feature:

$$\hat{y} = argmax_y(P(y)\prod_{i=1}^{n} P(x_i|y)).$$

   The Gaussian Naive-Bayes Classifier simply assumes the following in its estimation of the above quantity:

$$P(x_i|y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} exp(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}).$$

   Given that this is the simplest implementation of Naive-Bayes included in the scikit-learn package, it seemed to be a worthwhile inclusion, mostly to see whether the assumptions of conditional independence and a Gaussian distribution lead to a better-performing model overall [**?**].

4. Nearest Neighbor Classifier ("KNeighborsClassifier()" in scikit-learn)

   (a) The KNeighbors Classifier is another foundational approach to classification problems. Unlike the other models considered above, nearest neighbors is a kind of *instance-based learning*: instead of building a full model, it simply stores instances of the training data and uses them to predict a label [7]. By applying some kind of difference metric to the query point, it

can locate and identify the labels of the k "closest" points, and therefore assign it the majority class. This model was included in the suite as it is a very common approach to classification, and so we aimed to see whether it would hold up in these circumstances.

5. Multi-layer Perceptron Classifier ("MLPClassifier()" in scikit-learn)

   (a) The MLP Classifier is scikit-learn's implementation of a standard neural network. The model takes in an input and attempts to learn a non-linear function to predict its output by optimizing several internal layers. Due to its complexity, this approach is much more time- and computationally-intensive. In order to gauge whether it would perform noticeably better than our other models, however, we added it to the suite. It is worth noting that both the activation function used in the neural network and the learning rate were among the hyper-parameters tuned by the script; as such, the performance of this model may be very dependent on which internal functions it utilizes to develop the more general input-output function.

6. Support Vector Classifier ("SVC()" in scikit-learn)

   (a) The Support Vector Classifier is another fundamental model, build according to the design of a typical support vector machine. Support vector machines use sampled training instances called support vector to attempt to identify a decision boundary between two target labels; they often apply kernel functions to this data to more easily and dependably elicit a reliable boundary. This model was thus included in the suite both as a representative of the SVM approach and because of its versatility, due primarily to the many options in kernel for which one may select in the training process.

7. Bagging Classifier ("MLPClassifier()" in scikit-learn)

    (a) The Bagging Classifier is essentially a generalized version of the Random Forest Classifier. While it also bootstraps samples and subsets its feature sets, it can do so with any ensemble of models. One may select any combination of classifiers, whose outputs are aggregated at the time of prediction, as is the case with the random forest approach. In the case of this project's implementation, a combination of Decision Tree and Support Vector Classifiers were arbitrarily selected. It may be worthwhile applying different combinations of models to this particular ensemble in the future.

8. Stochastic Gradient Descent Classifier ("SGDClassifier()" in scikit-learn)

    (a) The Stochastic Gradient Descent Classifier applies the standard optimization process of the same name to a training set in the hopes of learning a linear classifier for the target label. Unfortunately, this classifier requires a number of bold assumptions with regards to its objective function; for it to function correctly, for instance, the objective function must be differentiable and convex. Despite these limitations, however, it is a very popular technique for learning a linear classifier, and we were interested in seeing how it performed in the case of our particular problem.

Integration

After developing each of these models, the next logical step was their integration into the final product. The chapter on "Evaluation," available below, offers a bit more detail about which models were selected for this process; this section will defer discussion of that subject until that point. Nevertheless, in its current form the application considers only the Random Forest Classifier, which was judged to be the best performing of the set. Upon execution of the appropriate script, this already-

trained model is applied to each scraped article in turn, before they are added to the Idioma database system. In the future, it may be worthwhile to explore other options on how best to integrate this modeling suite into the downstream application.

4.2.3 Database

The database utilized by Idioma is a group of collections hosted on a MongoDB Atlas cluster. MongoDB Atlas is a fully-managed cloud database created and maintained by the developers of the MongoDB system. Atlas seemed like the best platform for this project mainly due to its integration with the cloud; since I was planning on hosting both the server and the front-end on my localhost (as I did not plan to host the application online at this stage), I wanted to ensure that my database were located elsewhere. Furthermore, MongoDB Atlas was incredibly intuitive and free of charge!

In order to establish a connection between the above cluster and the application back-end scripts, PyMongo was needed. Essentially a wrapper for a Python-MongoDB tool-suite, this library proved to be very useful in sending information back and forth between the two endpoints.

The remainder of this section will provide a list of the collections (i.e. tables) used in the project, as well as their contents. They are shown in ***Tables 4.2***, ***4.3***, and ***4.4***.

4.2.4 Web Application

This section will consider the most tangible aspect of this project: the Idioma application itself. After providing an overview of a typical user's flow through the web app, it will discuss each screen and its corresponding back-end integrations in greater depth.

Let us begin by considering the architecture of the application itself. ***Figure 4.5*** provides a diagram demonstrating the connections between the screens of the

"users" - collection of Idioma users	
Field	Description
"_id"	a unique ObjectId identifying the user object
"f_name"	the user's first name
"l_name"	the user's last name
"email"	the user's email address
"documents read"	the number of documents read by the user
"sessions completed"	the number of times the user has logged in and out of Idioma
"password"	the user's password (Note: The current password storage system is insecure and should be updated in a future iteration.)
"achievements"	an Array containing the ObjectId of every achievement that the user has unlocked
"achievements_times"	an Array containing a timestamp for when the user unlocked each achievement; maps to the "achievements" Array index-wise
"documents_seen"	an Array containing the ObjectId of each document the user has already seen
"features_seen"	an Array containing an integer count of the number of times the user has seen each grammatical structure of interest to the document selection algorithm; the index-structure mappings are located on the back-end, and a list of these structures is provided in the "Web Application" section below
"level"	's' ("simple") or 'd' ("difficult"); the "experience level" of a user, corresponding directly to the classifier readability outputs

Table 4.2: The fields in the "users" collection.

"achievements" - collection of all unlockable Idioma achievements	
Field	Description
"_id"	a unique ObjectId identifying the achievement object
"name"	the name of the achievement
"desc"	a description of the achievement
"pts"	the point value of the achievement

Table 4.3: The fields in the "achievements" collection.

"featurized_diario_noticias_50iter" - collection of integrated documents	
Field	Description
"_id"	a unique ObjectId identifying the achievement object
"link"	a link to the document on the *Diário Notícias* website
"title"	the title of the article
"subtitle"	the subtitle of the article
"author"	the author of the article
"date"	the date on which the article was published
"text"	the body of the article
"tags"	an Array containing the tags that the author associated with the article
"features"	an integer Array containing an indicator of how many times a given grammatical structure or feature is present in the article - these counts are mapped index-wise to a feature dictionary present in the back-end scripts
"diff"	the "difficulty level" of the document, corresponding directly to the classifier readability outputs

Table 4.4: The fields in the "final_featurized_diario_noticias_50iter" collection.

application and how a user might move through them. As mentioned above, the Idioma front-end is built in React, a JavaScript library developed and maintained by Facebook that prides itself on facilitating component construction and sharing and on streamlining the DOM-rendering process. This framework communicates with the back-end, built using Python and Flask, and with Auth0, a third-party API used for secure authentication. Altogether, they provide the user with a clean and secure learning experience.

The user starts at the Idioma home page, pictured in ***Figure 4.6***. This screen displays a placeholder logo for Idioma (developed using Looka, an online logo creation tool) and a welcome message, set in front of one of four randomized background images. The home page aims to make the user's first impression of Idioma one filled with the beauty of Portuguese culture. As such, I selected four landscapes depicting picturesque destinations close to my own heart: the Alentejo coast, a vista near my father's home in Olhalvo, the central Portuguese plains, and the National Palace in

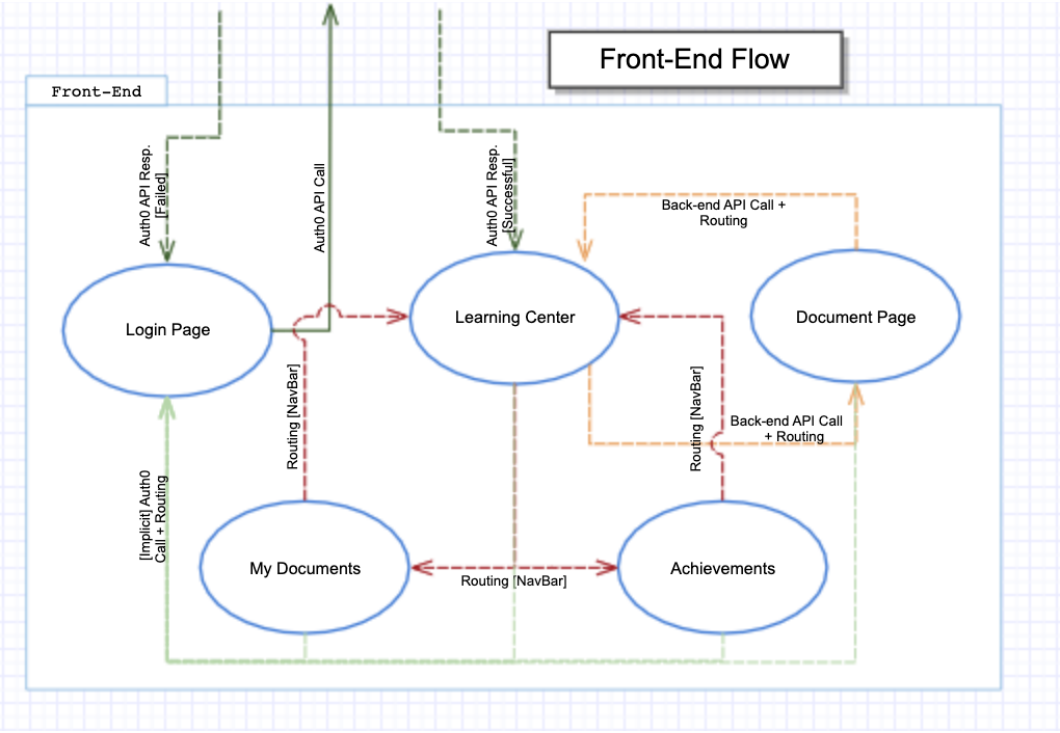

Figure 4.5: An architecture diagram detailing the typical user flow between screens in the Idioma application.

Sintra (pictured here). Future iterations of the project should extend these potential backgrounds to cover locales in other corners of the Lusophone world.

Upon clicking the "Log in!" button, the front-end launches a call to the Auth0 API, and the user is redirected to the login screen shown in ***Figure 4.7***. As is clear from the image, the screen is decorated with a few Idioma-specific details, thanks to Auth0's surprising customizability. Nevertheless, a few functionalities are yet missing from this page. Due to time constraints, a user cannot currently make a new account nor can they "Sign in with Google"; the application is currently limited only to the single user profile that we have created. That said, these should be fairly easy additions, and the current implementation is both stable and secure. The Auth0 platform appears to keep the user authenticated through a mixture of cookies and function hooks that trigger dynamically when tabbing between screens. Admittedly,

Figure 4.6: The home page for the Idioma application, depicting one of four possible randomized backgrounds.

the details are obscured a bit by the Auth0 API; thankfully, however, it does its job quite well.

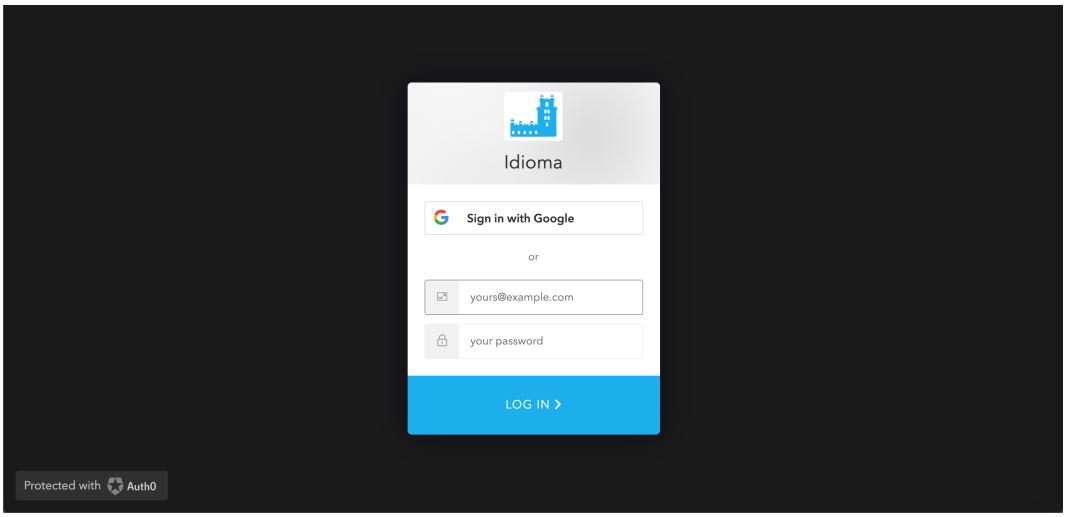

Figure 4.7: The login page created by the Auth0 platform.

That said, the existing user need simply enter their credentials and submit the form to be redirected to the Idioma "Learning Center" page, pictured in ***Figure 4.8***.

The Learning Center is the central nexus of the Idioma platform. On this screen, a user can navigate to one of the other available screens or begin the learning process by inputting the requisite criteria. Through the former of these options, the user can

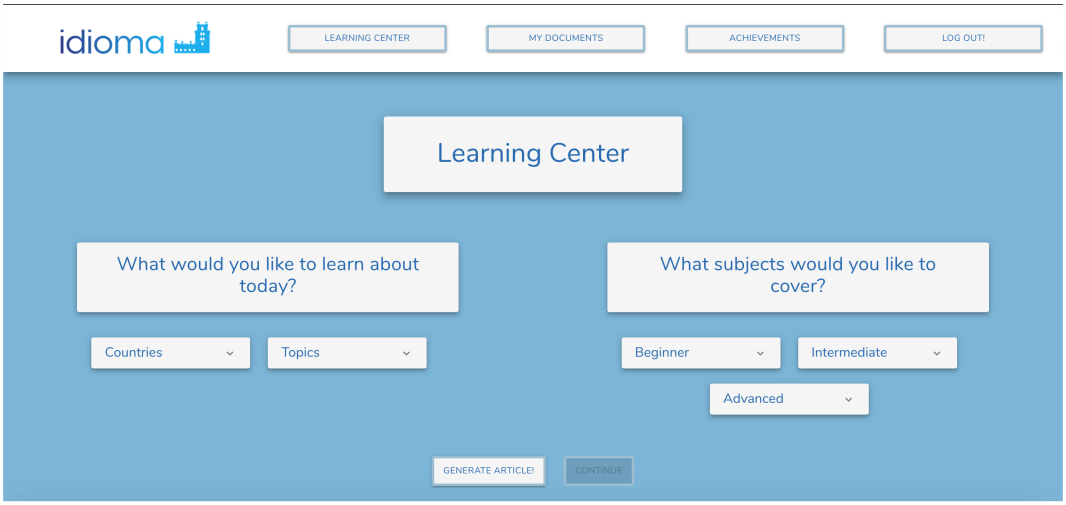

Figure 4.8: The Idioma "Learning Center," where a user can request a document to their specifications.

select one of the buttons located in the navigation bar at the top of the screen to route to the requisite page. Using React-Router, the application enables this navigation without forcing a full page refresh. This section will consider each of these other screens shortly.

The more critical function of this screen, however, is to enable the learning process. As is clear from the image, a user has two options for customizing their article selection: by content and by language structure. On the one hand, the user can select which thematic subjects they would like to cover. As this screen loads, the front-end sends a request to the '/api/getCommonTags' function, which returns a collection of the most common tags in the document database.[5] A user can toggle any of these tags, and the algorithm will weight documents of that subject matter more heavily during the curation process. Idioma also supports the latter feature: search by grammar. A user can expand the three right-hand dropdown menus, selecting any combination of features from them that they see fit. The algorithm will weight articles with these features more heavily as well, just as in the prior functionality. After

[5]We note that this communication is enabled by Axios, a Promise-based HTTP client that facilitates the transmission of HTTP requests and responses between an application's front- and back-ends.

selecting these criteria, the user may simply press the "Generate Article" button, and in so doing trigger a call to the API function "/api/getArticleId."

Now, when Axios successfully communicates this request and, therefore, the requisite parameters to the getArticleId function, it begins by querying the DB for the user object, extracting their level from that object (i.e. 's' or 'd') and retrieving all of the documents in the collection of that level. Next, it filters out those documents that the user has already seen by comparing the list against the "documents_seen" field in the user object. This leaves the function with a set of viable documents from which to select.

This selection turned out to fall among the most involved aspects of the project. In order to recommend a document that was a true "best fit" for the user, it was necessary to create an algorithm that would take into account both the grammatical features present in the article and the way in which those features interact with one another in the more general, linguistic sense. For instance, this algorithm should find the number of future-tense verbs in the document and select accordingly, but it should also consider the fact that knowledge of the present-tense is required to understand this more complex grammatical structure, and so refrain from recommending it to an individual who has not yet experienced this simpler tense. Thankfully, this problem became tractable with the help of a few Portuguese textbooks.

The first step was to decide on a feature set. These features would be the primary selection criteria; the algorithm would comb through the texts and extract these features, using them as the basis for this leg of the recommendation process. After consulting several Portuguese textbooks, we settled on the features listed in ***Table 4.5***.

Now, let us consider the process in stages. Firstly, before the function is even triggered, the developer must run all of the documents through a featurization process. After the articles are initially scraped, they are saved in the "Data" directory. The

Regular Verb Tenses	Irregular Verb Tenses	Misc. Features
Present Indicative	Present Indicative	Ser vs. Estar
Preterite Indicative	Preterite Indicative	Interrogatives
Imperfect	Imperfect	Superlatives
Past Subjunctive	Past Subjunctive	Prepositions
Present Subjunctive	Present Subjunctive	Numbers
Future Indicative	Future Indicative	
Conditional	Conditional	
Gerund		
Present Perfect		
Preterite Perfect		
Future Perfect		

Table 4.5: The features extracted from each text for selection.

developer must then manually navigate to the "Models" directory and invoke the script "featurize.py," which extracts the features needed for classification from each text, applies the modeling scheme discussed in the "Results and Analysis" section below, elicits and adds to the document object the features needed in the model selection algorithm (discussed shortly), and rewrites the data to a new JSON file. To perform the first task, a bit of language parsing suffices. To perform the second, however, a more heavy-duty approach is required. To extract the more grammatical features from the text, the script uses a framework called spaCy. An open-source library for advanced natural language processing, spaCy furnishes a number of pre-trained neural networks in a handful of languages, such as Portuguese. This script calls on the model to perform part-of-speech (POS) tagging on the text, labeling each token with what it belives to be its corresponding reference. The script then combs through this tagged text, counting up the number of times that each relevant feature surfaces, and saves it in the DB under the "features" field of the corresponding document object.

Another entity that must be created before the execution of the script is a dependency graph. In theory, this is a data structure that attempts to establish dependency relations among all of the grammatical features above; in practice, it is merely a hash

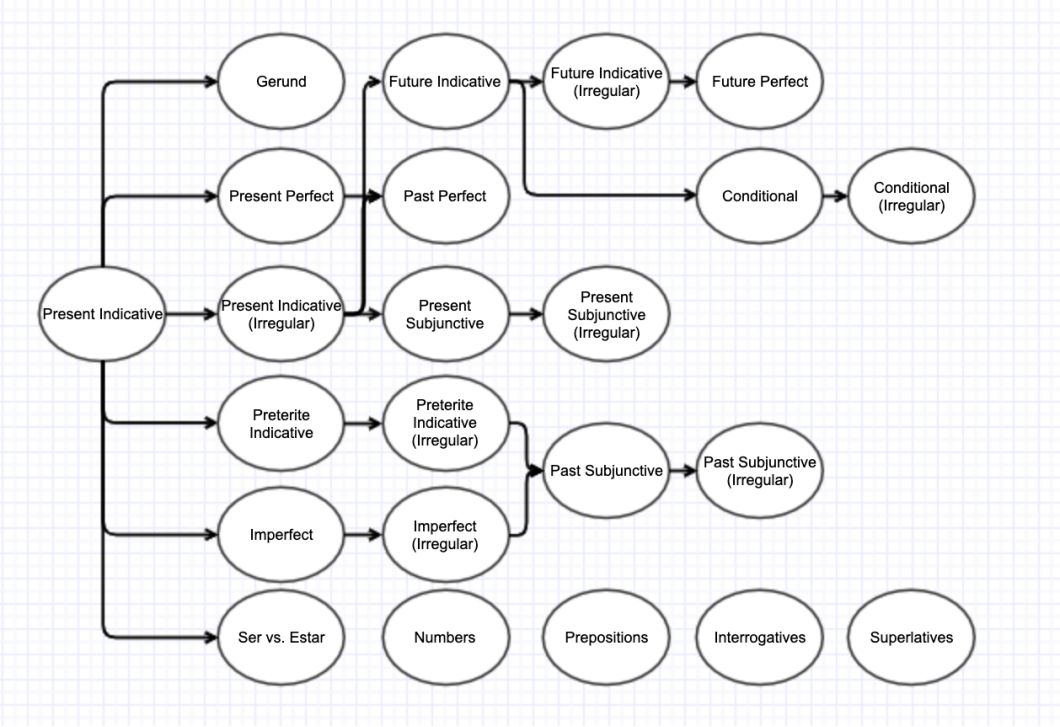

Figure 4.9: A flowchart depicting the dependencies between selection features.

table, where each key (an integer corresponding to a specific feature) has as its value a list of the other integers (features) upon which it is dependent. The index for "Present Indicative" would be in the list for the "Present Indicative (Irregular)" key, for instance, since one would not usually learn about the latter without first experiencing the former. ***Figure 4.9*** visualizes these relationships in their current state using a flowchart.

The next stage of the process is to determine the "relevance" of each topic in the above list to the user. The algorithm employed by the script is as follows: using the above dependency graph, the script visits each node (i.e. grammatical structure) in topological order, calculating its relevance (either -1 [irrelevant], 0 [neutral], 1 [relevant]) based on whether the user has requested to study it and on how familiar the user has become with it and with the topics upon which it is dependent. The relevance is easy to calculate for the feature that is dependent to no other topics (present indicative) or for any feature that the user wants to learn (i.e. toggles in the

Learning Center): a score of 1 is assigned. Otherwise, the script determines the score based on a conditional tree. If the user has seen every feature on which the query feature is dependent, and if they have seen this particular feature many (10+) times, then it is deemed to be neutral (0); if the latter condition is not fulfilled, then it is deemed to be relevant (1) in order to allow the user to practice this structure. Next, if the user has not seen any of the necessary previous topics, the query feature is deemed irrelevant (-1). Furthermore, if the user has seen at least 2/3 of the necessary previous topics and has not seen the query feature many (10+) times, it is deemed relevant (1); the thought process behind this decision is to expose readers to features that they may be slightly uncomfortable with, as long as they have enough requisite knowledge to make it work. If none of these conditions are fulfilled, the query feature is deemed irrelevant (-1). By performing these calculations on the feature in topological order, only one pass is necessary; as in functional programming, the requisite information for a present iteration has always been completed in a prior one.

Nevertheless, the script's task is not yet finished; it now needs to combine the relevance of each feature with its prevalence in each text, and output a score that ties these two criteria into a measure of the text's appropriateness for the user. It first samples 100 documents from the list of viable texts; it would be terribly inefficient to perform these calculations on all of the documents, and so we settle for the best of document of the sample, as opposed to the best overall. Then, for each text, it multiplies the relevance of each feature (-1, 0, or 1) by the normalized prevalence of each feature in that articles (0, 0.5, or 1), adding all of these products together, and then dividing by the number of features. This transformation should create a normalized score for the text, ranging between -1 and 1, which can be used for objective comparison among all of the candidates. Now, the script also adds a bonus modifier to a document's score if it has any of the user's desired subjects in its tags. The value of this bonus is configurable within the script itself, but we settled on a

Figure 4.10: An example Idioma document page.

value of 0.1 to start. Now, the current implementation of these scripts simply selects the document ID with the highest corresponding appropriateness score. Once the choice has been made, the script returns the ID to the front-end for display.

Once this operation has completed, the greyed-out "Continue" button in the Learning Center should become clickable; selecting that toggle navigates the user to another page, upon which the document is pictured, as in ***Figure 4.10***.

Now, this screen is fairly self-explanatory; it is merely intended to show the article to the user for their viewing pleasure. When the page begins to render, it supplies the article ID (passed in as an argument using React Router) to an Axios-driven back-end call to "/api/getArticleData," which retrieves all of the data for the corresponding article. The title, subtitle, link, text, and other information is all displayed within a component on the page for the user's perusal. Now, once the user returns to the Learning Center via the button at the bottom of the page, another call is made to a separate API function: "/api/updateUser." This function updates the users collection in the Atlas cluster with the number of new occurrences of each individual feature that the user has just experienced; if the article contains five interrogatives, for example, the corresponding entry in the user's "features_seen" field is incremented by five and

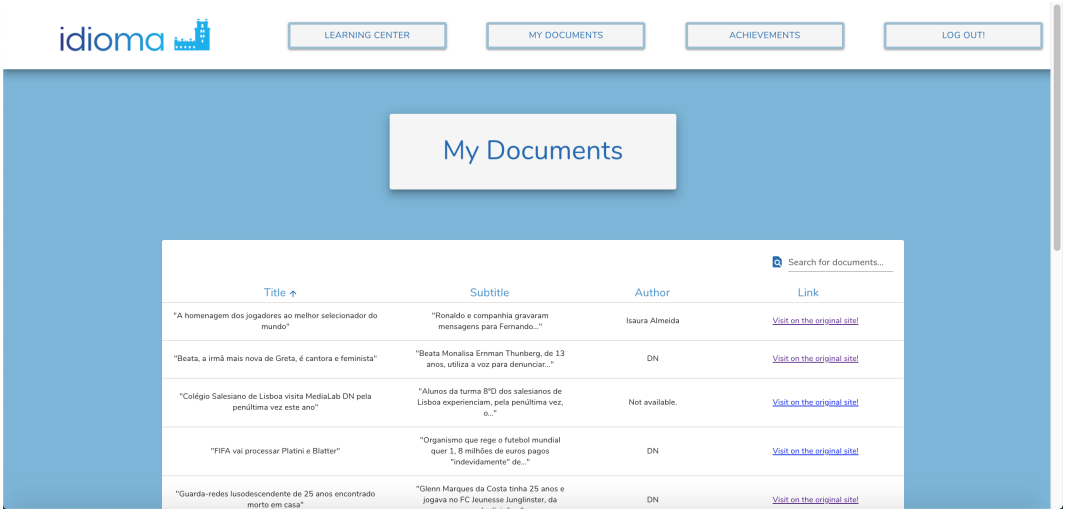

Figure 4.11: A user's "My Documents" page in the Idioma application.

refreshed. This detail allows the application to learn in lock-step with the student; as the student grows and experiences more of these grammatical structures, their progress is reflected in their profile and the algorithm begins to favor other, more appropriate documents. By putting all of these pieces together, Idioma manages to achieve one of our primary objectives for this project: dynamic, customizable learning.

Moving on, the application offers a handful of other screens that are also important to this mission in their own right. The first of these is the "My Documents" screen, accessible by the navigation bar and pictured in ***Figure 4.11***.

It seems unlikely that any student will have fully digested a new article after the first read; furthermore, that same student may want to revisit an article that they read for any number of reasons, perhaps to test themselves later on in their learning experience. For this reason, Idioma contains the "My Documents" page, a database of all of the documents that a given user has read throughout their time on the application. As is clear from the figure, the screen contains a searchable, sortable table, which itself lists all of these articles by title, subtitle, and author; this table relies on a number of straightforward back-end functions to supply it with this information, but details are omitted here for space considerations. A user can choose

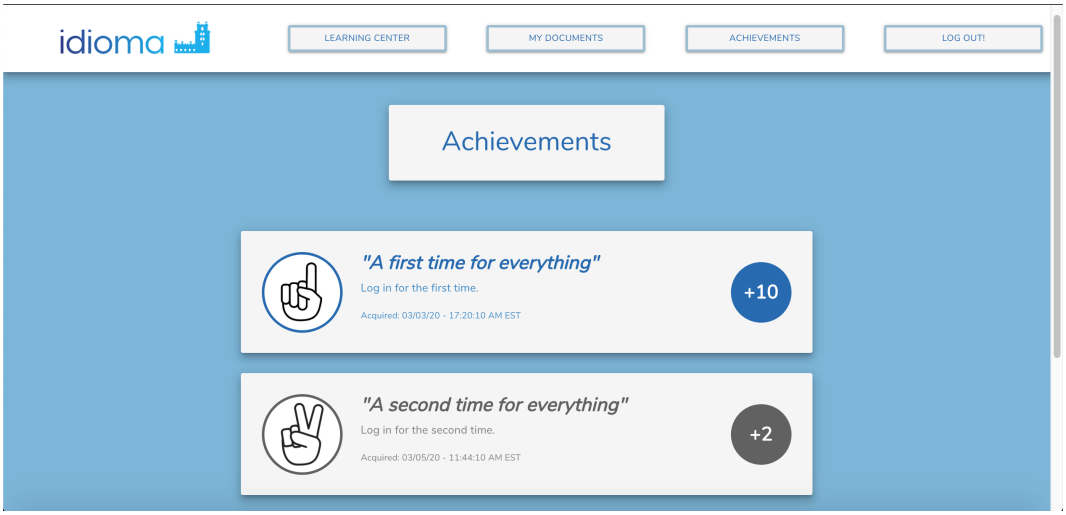

Figure 4.12: A user's "Achievements" page in the Idioma application.

to read the article on the original site (a detail that had the added bonus of giving further credit to the authors), or navigate to an altered of the document page on Idioma by simply clicking on the body of the table entry. Idioma is meant to be a tool that a student can revisit indefinitely, and it was our hope that this feature would push the application in this direction.

Another feature that was intended to keep the user coming back was the Achievements Page, similarly accessible via the navigation bar and depicted in ***4.12***.

In addition to providing the student with great learning functionality, we also hoped to make Idioma a fun application to use. One way to achieve this objective was through gamification, specifically through achievements. Achievements are essentially digital, score-driven trophies that a user unlocks after meeting a specific criteria. These systems have been fully realized in countless platforms, from video games to other language-learning applications, and Idioma should be no exception. The Achievements Page allows a user to view their progress in this field. Upon rendering, the screen launches the "/api/getUserAchievements" and "/api/getAllAchievements" functions, cross-referencing the user object's "achievements" field against the list of all potential achievements. When this is done, the front-end displays each achievement

in the latter list in an organized table, differentiating those that the user has unlocked from those that they have not by painting them in color and in grayscale, respectively.

Now, the implementation of these achievements differs by their content. At the time of writing, three achievements have been implemented: "A first time for everything," "A second time for everything," and "Beginner." The former two achievements track when a user has completed their first and second learning sessions (i.e. logged in and out of Idioma for the first and second times), while the third is unlocked after the user reads their first document on the platform. The general implementation structure for each of these achievements is the same: when the user takes an action that could result in an achievement unlock (i.e. logging out, or tabbing out of a recently-read document), an API function is called that updates the corresponding fields in the "users" DB table and checks their criteria for the corresponding achievement. If, for example, a user completes their first session when they press the "Logout" button, an API function will make note of this by conferring with the Atlas DB. If the user action successfully triggers an unlock, the user profile is updated and an alert is shown congratulating the user. This new achievement is then visible on the aforementioned Achievements screen. Though only three achievements have been implemented thus far, it is straightforward to add more, and so this presents an excellent opportunity for future contribution.

Nevertheless, it was our hope that these achievements would work in tandem with our proprietary, customizable selection algorithm and with the persistence of the user's document history to create a fun learning experience that would incentivize the student to keep coming back to Idioma. Regardless of how it compares to other, comparable platforms, it certainly seems to check those boxes!

Chapter 5

Evaluation

This chapter will provide a overview of our methods for evaluating the models trained for Idioma's classification problem. After a brief discussion of the methods employed, which rely heavily on the scikit-learn libraries noted above, it will reflect on the results of these processes and on how they influenced the task of model selection.

5.1 Methods

After training each of the eight models in the suite, the Python scripts immediately begin the evaluation process. Before the training process had begun, the scripts used the scikit-learn "train_test_split" function to independently and randomly divide the data into training and testing sets, of sizes 80% and 20% of the size of the larger data set, respectively. At this stage, each model is used to predict the labels of each instance in this latter set, after which several metrics are derived to measure its performance.

Let us briefly consider these metrics, which number four in total: accuracy, precision, recall, and f1 score. Accuracy is the simplest of the group; it is merely the total number of correctly-predicted instances over the total number of instances. The latter three metrics provide us with more nuanced reflections of the model's strengths

and weaknesses. Precision, for one, is the proportion of "true positive" predictions over the total number of positive predictions. Now, we note that, in the case of these classifiers, a prediction of 'd' (or, "difficult") is considered to be positive; this was simply a notational choice, as the developer must select one of the two labels when performing a binary classification. Nevertheless, the precision of the model provides a measure of how many positive predictions actually prove to be correct, an essential metric when preventing false positives.

A kind of foil to the precision metric, recall calculates the proportion of true positives to the sum of all true positive and false negative predictions. This expression essentially captures the fraction of positive instances that the model actually predicts correctly. Unlike precision, this metric prioritizes the prevention of false negative predictions (in this case, "difficult" documents classified as "simple"). The f1-score, on the other hand, takes both of these metrics into account by calculating their harmonic mean. Essentially a more complex version of the accuracy metric, the f1-score is especially useful when there is an uneven target class distribution among the instances.

Now, the evaluation process will take all four of these metrics into account, but it is worth considering which are the most useful in this application. Between precision and recall, for one, the latter is likely more important. The reason for this assessment is the argument that a false negative is more damaging than a false positive in the context of Idioma. If a user suffers a false positive, they are simply provided a document that is considered difficult but, in reality, proves to be fairly straightforward; aside from a few moments of confusion at the algorithm, there is not much at stake here. If they suffer a false negative, however, they are supplied with a "simple" document that is actually quite challenging. While a certain amount of difficulty is built into the selection algorithm, such a spike could put the reader off from the learning process

altogether. For this reason, we weight recall a bit more highly in the evaluation process than precision.

Furthermore, both accuracy and the f1-score should arguably be weighted less than both precision and recall. The reason for this is that they essentially provide simple averages, whereas precision and recall offer more nuanced assessments of the model's high- and low-points. We note, as above, that the f1-score is most useful when there is a target class imbalance in the data set. But since the training data is fairly balanced (about a 60-40% split between 'd' and 's' classes), this metric is not quite as powerful as it could be in another, comparable application. For this reason, we consider precision and recall to be more relevant than accuracy and f1-score, with the latter in each pair given a bit more weight than the former.

5.2 Results and Analysis

The results for the above evaluation process are plotted in ***Table 5.1***. For each criterion, the highest-scoring value was circled in red while the two runners-up were circled in black (in the case of Precision, third place was shared among Bagging and k-Nearest Neighbor).

Metrics and Statistics	% Features Used	Accuracy	Precision	Recall	F-measure
Decision Tree	0.6	0.73	0.68	0.69	0.68
Stochastic Gradient Descent	0.65	0.55	0.49	0.98	0.65
Multi-Layer Perceptron	0.5	0.81	0.72	0.88	0.79
Gaussian (Naive Bayes)	0.5	0.7	0.8	0.35	0.48
Bagging	0.35	0.79	0.74	0.74	0.74
Random Forest ☆	0.55	0.85	0.79	0.84	0.81
Support Vector Classifier	0.45	0.79	0.71	0.86	0.78
k-Nearest Neighbor Classifier	0.65	0.8	0.74	0.81	0.77

Figure 5.1: The percentage of features used, accuracy, precision, recall, and F-measure for each implemented model.

Before we determine which of these is the "winning" model (or, perhaps, whether multiple classifiers satisfy the necessary conditions), let us first consider a few notable results. Firstly, it is curious that the Stochastic Gradient Descent model has very high recall (0.98) and very low precision (0.49, less than the probability of randomly guessing correctly). This disparity that the model experienced exceptionally few false negative predictions, as the recall figure is very close to 1, and more false positives than it did true positives, as the precision is below one-half. According to our criteria, this pattern does not immediately invalidate the model; after all, we should prioritize recall over precision, and this classifier is somewhat of an extreme exemplification of that division. However, it is frankly unacceptable to utilize a model that incorrectly classifies as difficult more instances than it correctly classifier as difficult, and so these metrics are a major strike against Stochastic Gradient Descent.

A similar, yet inverse, pattern can be found in the Naive Bayes classifier. Despite having the best precision score, its recall is abysmal, with only about one-half as many true positives as it has false false negatives. It seems to be the case, then, that the Naive Bayes model is very competent at finding documents that are actually difficult; perhaps it is able to identify key features that denote a greater challenge, such as a specific cutoff in one of the readability scores or otherwise. Given that this kind of classifier makes quite bold independence assumptions in its prediction process, it may be the case that they do not hold often enough to create this disparity in performance. Whatever the case, it is clear that these metrics do not paint the Naive Bayes model as a strong candidate for Idioma.

Lastly, it is also interesting to note the percentage of features used by each of these models in the training process. While this proportion appears to oscillate around 0.5-0.6, Bagging is a clear outlier. This classifier utilizes only 35% of the candidate features in making its predictions. And this is not to its detriment; although it does not excel in any one particular metric, it performs fairly well across the board.

The question remains, however, of why this is the case. We postulate that this trend can be attributed to the fact that bagging is an ensemble model, aggregating the results of several individual classifiers to predict a label. As such, too many features could potentially lead to overfitting on the training data; though bagging employs bootstrap sampling and random feature selection amongst member models, this could be an extra precautionary step that it takes to ensure that its results can be generalized. This conclusion appears to be corroborated by the low proportion of features used by the Random Forest model, another ensemble method. Nevertheless, it would be interesting to investigate this trend further to see if it applies to other ensemble methods outside of those tested here.

Now, let us consider which we determine to be the best-performing or most appropriate models for our application. The "winning" models (i.e. those marked with a red star) were judged to be the Random Forest, Multi-Layer Perceptron, and Bagging models. To arrive at this conclusion, we considered the results from a number of perspectives. We first took note that the Random Forest classifier had the highest number of 1st-place metrics (2), which it and the Multi-Layer Perceptron classifier had the highest number of metrics place in the top 3 (3). These observations immediately established these two models as front-runners. This assessment was further corroborated by the fact that the latter showed the highest recall, the former showed the highest precision (outside of Naive Bayes, which was determined to be inappropriate for our purposes above), and they hold the top 2 f1-scores amongst all of the candidates. These reasons led us to consider these two models to be the most obvious candidates to implement.

In a more simplistic application, it would suffice to simply choose one of the two and integrate it. In order to get a diversity of voices in the selection process, however, we believed that it would make sense to use a best-of-3 approach; rather than relying on the labeling of only one model, we would integrate three distinct classifiers and

take the simple majority of their outputs. Clearly, then, a third model was necessary to fill out the suite, for which purpose we selected the Bagging classifier. Rather than specialize in any one of the metrics, Bagging performed fairly well in all of them; we believed that this kind of implicit impartiality would help to balance the 3-model suite, since Random Forest and MLP are both skewed a bit in the direction of Recall.

Thus, we are left with a collection of three distinct models that help Idioma to provide consistent, appropriate documents to its end-users.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

In its current state, Idioma provides an excellent user experience that achieves all of the objectives that we deemed necessary to fill the void left by current available language-learning applications. On the one hand, its reliance on natural language processing and machine learning techniques satisfied our desire for a cutting-edge solution; though many comparable services leverage cutting-edge tech, Idioma's recent development window and direction allowed it to make use of burgeoning models and frameworks in its delivery of a state-of-the-art user experience. On the other hand, the application also delivers on the promise of a dynamic, customizable, reading comprehension-driven platform. The ability to tailor the reading content allows the user to take charge of their learning, while the proprietary algorithm constantly working on the back-end ensures that the service stays up-to-date with a user's strengths and weaknesses. All the while, students get the chance to experience these features through an expansive database of articles, varying in difficulty and in subject matter, that give them the practice in reading comprehension that they need to truly become proficient in the Portuguese language. Despite its limitations, there truly does not

exist a service like Idioma on the market, and we are very proud to have delivered the foundation to this unique premise.

At the time of writing, all of the code related to Idioma and its constituent services is available for download in a publically-available GitHub repository. The link to this repository is as follows: https://github.com/paulo892/IdiomaFinal. It is our hope that future contributors will be able to help address the many opportunities for future work that are currently available, which this paper will now detail.

6.2 Future Work

While Idioma's suite of features is larger than even we had expected at the outset, there yet exist a wide variety of opportunities for future contributions and extensions to the project.

Let us first consider those elements that are external to the application itself, yet critical to its proper functioning: the web scraper and the modeling suite.

The web scraper offers one exceptionally clear opportunity for future work, namely due to the fact that it is presently limited in scope and easily extensible. At the time of writing, the web scraper was tailored only to one journalistic website: *Diário Notícias*. In theory, these scripts could be adapted to perform the same functions on any website, adding a greater variety of voices and subjects to the Idioma document database. Furthermore, the web scraper was last run in mid-March; therefore, the documents that a current user would see are actually fairly out-of-date, at least from a current events perspective. Therefore, any interested contributor could take on the task of extending the web scraper to pull from other sources and to do so more frequently, to ensure a more current pool of documents. They might even add a new functionality to allow the user to request specific websites, or even pull the data themselves on the fly; this would confer upon the user an even finer level of control

over their learning. In practice, however, this task is not as easy as it appears. While very user-friendly, Scrapy is not a trivial library to develop upon; it is very dependent on the individual intricacies of a webpage's HTML structure, and so the inclusion of other sources in its execution path would require a fair amount of development time. This is compounded by the tendency of these news sites to use moderately complex JavaScript on their pages, which necessitates the use of Selenium requests on the part of the developer to gather the necessary data. These details make the scraping process more challenging than expected, and make us more confident in the success of our existing application.

The modeling suite is another area ripe for further contribution. Though not a limitation, as we believe the best-of-three approach that arose from the modeling suite performs quite well, it would be straightforward for an individual to conduct their own research and implement new additions as they see fit. Machine learning is a burgeoning field, and the current literature is constantly being extended to include new permutations of older models as well as completely novel approaches. This component in the Idioma platform is limited insofar as it sources only from those techniques that are viable today, as well as those implementations available for adaptation in the scikit-learn library. In addition to adapting new breakthroughs in binary classification, an individual might simply take the time to incorporate models for which scikit-learn does not yet offer a suitable implementation. Though this would take a fair amount of time, especially if these models have to be constructed from scratch, it could be a worthwhile opportunity for future extension.

Similarly, the data used to train these models could also be further refined. We are incredibly grateful to Filho for offering us his data set; without his assistance, this project would be nowhere near complete today, and it actually works very well in using only the data that he provided. Nevertheless, the target labels that this data set offers, namely 'd' for "difficult" and 's' for "simple," are not as useful as we had

hoped. It is challenging to determine whether a user is ready for difficult documents, as this classification does not follow a universal standard that can be easily calculated for a given student. A future contributor could, therefore, take it upon themselves to craft a data set that labels documents according to CEFR guidelines, which would enable the developer to simply compute a user's proficiency level according to those metrics and offer a more accurate learning experience. Another limitation pertaining to the data is the absence of featurization methods for syntactical and POS attributes in a candidate text. We were not able to develop a module to efficiently break down a sentence into its constituent parts, a task that would have enabled us to utilize the majority of features in the data set provided by Filho. The addition of such a module could therefore have a tremendous effect on the accuracy of the modeling suite.

Let us now move on to the application itself. This version of Idioma has a fairly extensive feature set, with each of the constituent functionalities working as expected. That said, there are a few opportunities for future work ready to be tackled. The first is the process of hosting the application. Initially, it was our goal to launch the Idioma alpha on a publically-accessible domain. Due to time constraints, however, this became infeasible. Given how user-friendly services like Heroku have become, however, it should be straightforward for one to migrate the front- and back-end onto this service.

The authentication system offers another occasion for extension. Though it can successfully authenticate a user whose credentials have been supplied to the Atlas DB, the portal is not configured to use Google Sign-in, nor can it provision accounts for interested new users. In addition, the current solution for storing user credentials (i.e. plain-text in an Atlas cluster) is frankly insecure; it would require at least some kind of hashing and salting algorithm to provide the user with security, which is not currently the case. These additions would be welcome additions to the platform.

Additionally, the User object within the application proper could be fleshed out as well. For one, there is currently no system in place for constructing a profile for a new user. We considered the idea of introducing some sort of quiz for a new student to take, which would provide some kind of feedback to inform the construction of their profile. Due to time constraints, however, we were unable to develop such a process, and the users presently included in the Atlas cluster were configured essentially at random. The User object is also limited by the lack of an algorithm to determine the appropriate skill level (i.e. 's' or 'd'). As noted above, the ambiguity of these categories makes the division of users between them a fairly difficult problem. As such, a future contributor could pursue a method to reliably do just that, or attempt to revise the training data such that the classification rests on more easily-differentiable target labels.

One final opportunity is more algorithmic in nature. Once again due to time constraints, we were unable to develop the feature allowing users to request documents based on their subject matter. In theory, this would be a straightforward addition. Along with the article text and metadata, our web scraper retrieves a list of tags describing the material as well. A handful of these tags actually correspond directly to a set of larger categories listed on the *Diário Notícias* homepage, while the others are more free-form in nature. Nevertheless, one could simply collate these descriptors into a dictionary of categories, from which a user could select the most interesting for their learning.

That being said, we are incredibly happy with the foundation that has been laid through this project. Though incomplete, Idioma achieves the objectives that it set out to accomplish, providing a high-tech, flexible solution to the problem of reading comprehension in Portuguese second-language learning. We are confident that, with a bit of extra time and care, Idioma could become a true asset for industrious students around the world (especially those who are socially distancing).

Bibliography

[1] BRWaC. https://www.inf.ufrgs.br/pln/wiki/index.php?title=BrWaC. [Online; accessed 19-April-2020].

[2] Business portuguese. https://www.memrise.com/course/1172079/business-portuguese/. [Online; accessed 19-April-2020].

[3] The CEFR levels. https://www.coe.int/en/web/common-european-framework-reference-languages/level-descriptions. [Online; accessed 19-April-2020].

[4] Emotions in portuguese. https://www.memrise.com/course/127669/emotions-in-portuguese/. [Online; accessed 19-April-2020].

[5] Memrise. https://www.crunchbase.com/organization/memrise#/entity. [Online; accessed 19-April-2020].

[6] NAEP reading 2019 highlights. https://www.nationsreportcard.gov/highlights/reading/2019/. [Online; accessed 19-April-2020].

[7] Nearest neighbors. https://scikit-learn.org/stable/modules/neighbors.html#classification. [Online; accessed 19-April-2020].

[8] Os animais. https://www.memrise.com/course/134461/os-animais/. [Online; accessed 19-April-2020].

[9] When duolingo was young: the early years. https://vator.tv/news/2018-06-22-when-duolingo-was-young-the-early-years, Jun 2018. [Online; accessed 19-April-2020].

[10] Fabio Crestani. *Experimental IR meets multilinguality, multimodality, and interaction: 10th international conference of the CLEF Association, CLEF 2019, Lugano, Switzerland, September 9-12, 2019: proceedings*. Springer, 2019.

[11] Pedro Curto, Nuno Mamede, and Jorge Baptista. Automatic text difficulty classifier - assisting the selection of adequate reading materials for european portuguese teaching. *Proceedings of the 7th International Conference on Computer Supported Education*, Dec 2014.

[12] Jorge Alberto Wagner Filho, Rodrigo Wilkens, and Aline Villavicencio. Automatic construction of large readability corpora. *Proceedings of the Workshop on Computational Linguistics for Linguistic Complexity (CL4LC)*, page 164–173, Dec 2016.

[13] Bente E. Hagtvet. Listening comprehension and reading comprehension in poor decoders: Evidence for the importance of syntactic and semantic skills as well as phonological skills. *Reading and Writing: An Interdisciplinary Journal*, 16:505–539, Sep 2003.

[14] Theodore L. Harris and Richard E. Hodges. *The Literacy Dictionary: The Vocabulary of Reading and Writing*. International Reading Association, 2005. Pg. 39.

[15] Sarah Logan, Emma Medford, and Naomi Hughes. The importance of intrinsic motivation for high and low ability readers reading comprehension performance. *Learning and Individual Differences*, 21(1):124–128, 2011.

[16] Danielle S. McNamara. *Reading Comprehension Strategies: Theories, Interventions, and Technologies*. Taylor and Francis, 2012. Pp. 244-249.

[17] P. Reitsma and Ludo Th. Verhoeven. *Problems and interventions in literacy development*. Springer, 2011. Pg. 207.

[18] Steve Ridout. About Readlang. `http://blog.readlang.com/about/`. [Online; accessed 19-April-2020].

[19] Tim Seeger. Lingua. `https://lingua.com/`. [Online; accessed 19-April-2020].