Called to Learn: Word Recommendation for Spanish Language Learning Missionaries

Lillian Bartholomew, Connor Franckowiak, Jonathan Merrill, Reece Robertson, Ben Summers

December 19, 2021

Abstract

The Church of Jesus Christ of Latter-day Saints has created an app to help its missionaries learn foreign languages. Our objective is to use machine learning to create a model that will recommend new words to missionaries learning Spanish based on the words that they have already studied. We test two potential models and recommend the best one for use in the app. The code used in this project can be found at the following link:

https://bitbucket.org/reecejr/vol3project/src/master/

1 Background and Motivation

The Missionary Training Center uses a web-based app called Embark to assist missionaries in the process of learning a foreign language. This app is open to the public and can be used by anyone with an account at churchofjesuschrist.org via tall.global/embark. Tasks within the app are focused around the missionary purpose. For example, one of the first tasks recommended to new users is "Offer A Prayer," one of the missionary lessons. These tasks are recommended to the user on the home page of Embark based on task type and whether the user has recently engaged in the task.

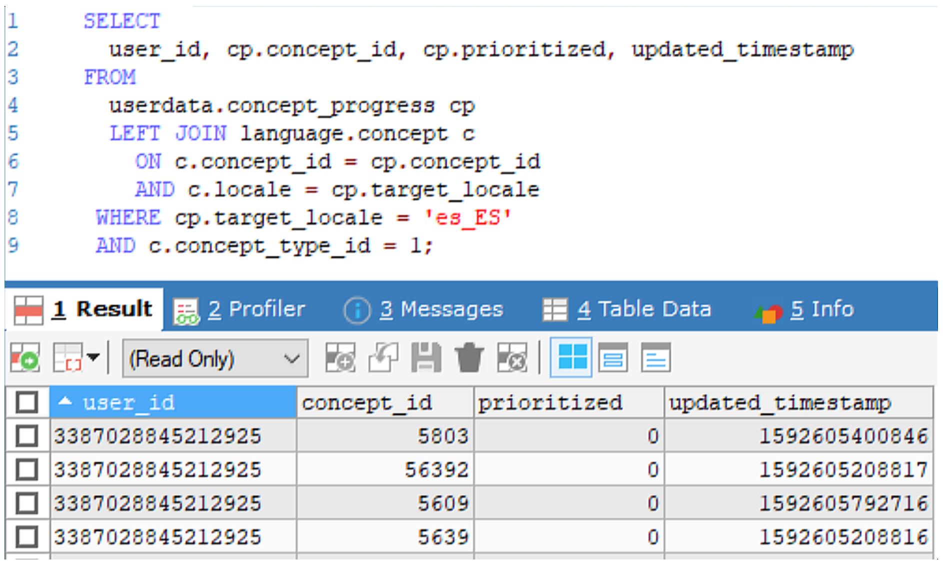
At the request of the manager of one of the researchers who is currently employed at the MTC, the goal of our project is to create a similar recommendation system for vocabulary concepts. We hope to create a model that will streamline the learning process by tailoring specific new words for the user given his or her current progress in learning the language, rather than recommending an entire task. By recommending specific words based on past language learning history, we hope to recommend words most useful to the user. For example, if a user struggles with consistent sound patterns, we hope to identify words with those patterns that they should study. Such a feature will also be useful for missionaries with limited time, as they can enter the app and study individual words curated for them rather than start, and potentially not complete, a given task.

Furthermore, one of the features of Embark is the ability for users to mark a word as a "favorite" with a star, indicating that they would like to study the word further. We explore predicting words that users might star in hopes to encourage them to gain a better understanding of key concepts, and spend their time on the most useful topics.

For simplicity, we restrict our analysis to the Spanish language, although the methods we employ below are easily applied to additional languages, with minimal modifications.

2 Data Collection/Cleaning and Feature Engineering

The data for our project came from a snapshot of anonymous user data taken on October 25, 2021, which was done under the permission of the aforementioned manager who provided the idea for the project. Due to the large size of the dataset, we ran a SQL query (see Figure 1).



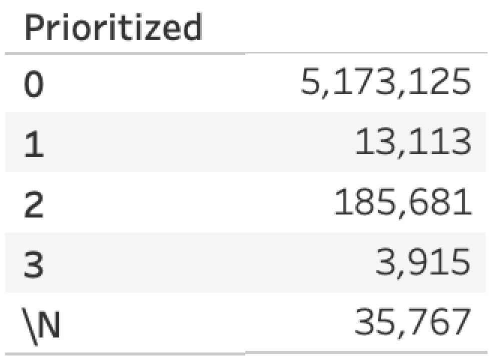
```sql
SELECT
  user_id, cp.concept_id, cp.prioritized, updated_timestamp
FROM
  userdata.concept_progress cp
  LEFT JOIN language.concept c
    ON c.concept_id = cp.concept_id
    AND c.locale = cp.target_locale
 WHERE cp.target_locale = 'es_ES'
 AND c.concept_type_id = 1;
```

user_id	concept_id	prioritized	updated_timestamp
3387028845212925	5803	0	1592605400846
3387028845212925	56392	0	1592605208817
3387028845212925	5609	0	1592605792716
3387028845212925	5639	0	1592605208816

Figure 1: The SQL query and sample of data

The initial data we retrieved consisted of 5,411,601 rows. Each row contained a single data point storing an individual user's history with a given vocabulary concept, as described by the following four features:

1. `user_id`, a 16-digit integer uniquely identifying a user.
2. `concept_id`, an integer corresponding to a vocabulary word.
3. `prioritized`, an integer between 0 and 3 that Embark uses to track whether the user has ever starred and/or incorrectly recalled the word during their use of the app. 0 indicates a word that has been neither, 1 indicates a word that has been mistaken but not starred, 2 indicates a word that has been starred but not mistaken, and 3 indicates a word that has been both.
4. `updated_timestamp`, a Unix timestamp in milliseconds representing the last time the user interacted with the concept.



Prioritized	
0	5,173,125
1	13,113
2	185,681
3	3,915
\N	35,767

Figure 2: The values of the prioritized column for the raw data.

Through examination of these features, we found that there were five different prioritized values instead of the desired four (see Figure 2). To remedy this we removed null values from the prioritized column, giving us the four desired values. We also removed the null values in the `updated_timestamp` and `user_id` columns and converted the `updated_timestamp` column to integer values. Using the time stamps, we created a new feature for the data called `words_studied`. This feature recorded the number of words the user had studied at the time they last encountered each word, thus it is a rough indicator of the user's progress over time. Then we augmented the `prioritized` column by splitting it into two Boolean features: `starred`, which is `True` for a word with priority 2 or 3, and `mistaken`, which is `True` for a word with priority 1 or 3. Both are `False` otherwise.

Finally, we noticed that a large percentage of users do not star words (as indicated by the high value of 0 in Figure 2). As can be seen in the difference

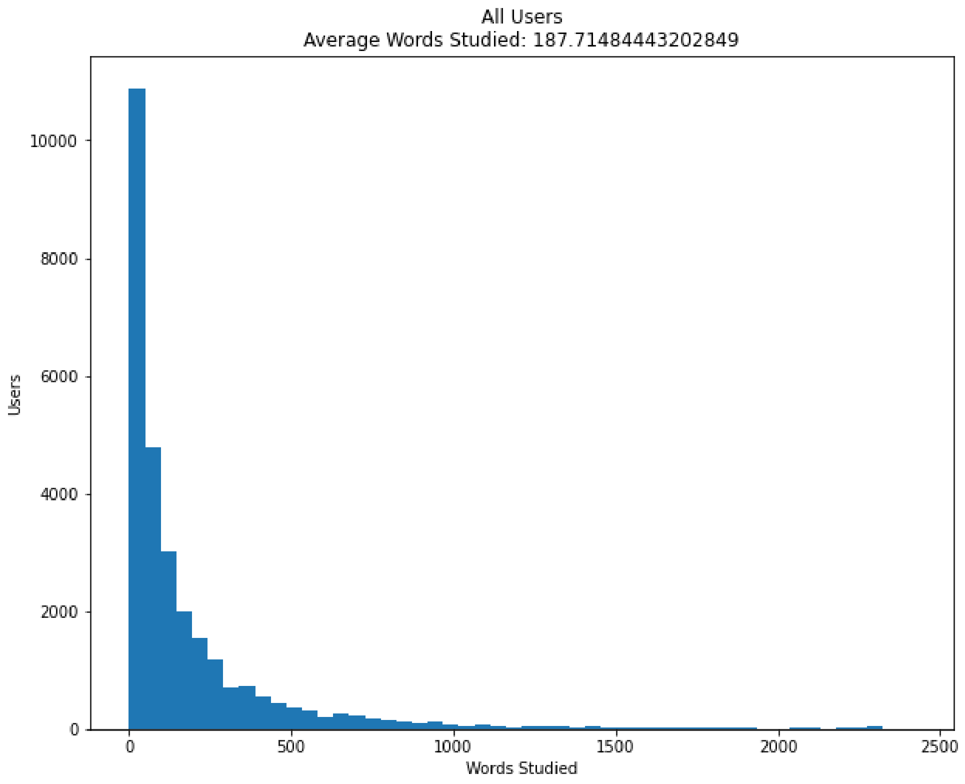


Figure 3: Words studied by all users.

between the number of words studied by all users (Figure 3) and the number of words studied by users who take advantage of Favorites (Figure 4), this second group of users is more invested in their learning experience. In an attempt to create an experience that is geared towards those who will benefit the most from recommendations, and to limit the size of the dataset, we filtered out all users that had not starred any words. This reduced the size of our data to 1,993,285 rows, roughly 40% the size of the original dataset. All data cleaning code can be found in `data_cleaning.py` in the BitBucket repository linked above.

3 Analysis and Results

After cleaning the data, we began working on models to recommend new words to users. We made two attempts, one using a Random Forest classifier,

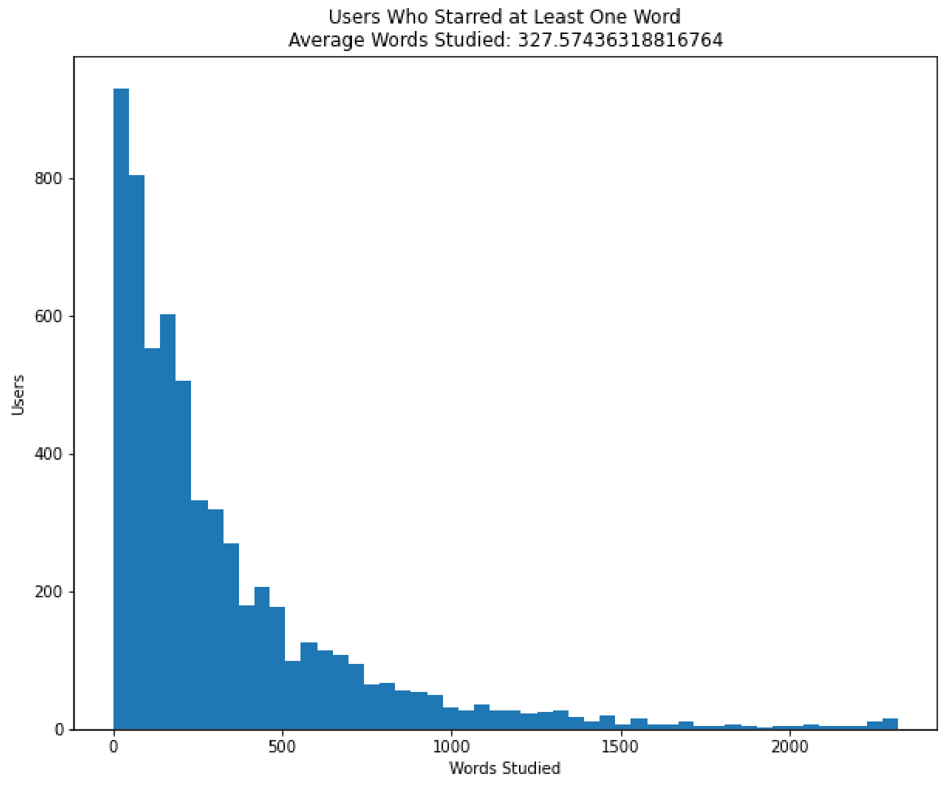


Figure 4: Words studied by users who starred at least one word.

and one using a K-Nearest Neighbors classifier. We also developed a metric to evaluate the efficacy of recommended words made by both models to give them comparable scores. This scoring function, as well as both models, are outlined below.

3.1 Scoring Function

Our project is not a typical classification problem. Rather than taking input data and predicting a class value, we are attempting to take in input data and output related data points. In other words, rather than building a model that predicts a class based on some given features, we want a model that is good at associating similar features. As part of this problem, we needed to develop a unique scoring function to evaluate the quality of predictions made by our model.

We split our data into a training set and a testing set by user. We used the training set to fit each model as usual, but we further divided our testing set into two pieces. Sorting each user's data chronologically, we used the first half as "recommendation" data, which is to say that we passed the first half of the user's data into each model to generate word recommendations. We then compared these recommendations to the second half of the user's data which was not passed into the model. Any word that was recommended which appeared in the user's subsequent activity was counted as a success– the model successfully offered a word that the user later studied. Each recommended word that did not appear in the user's later activity was not counted as successful. Finally, we averaged these successes and failures to get a total score for the model.

This scoring function is only a stand-in for the true measure our model's accuracy. The accuracy ought to, and will eventually be, determined by user feedback which reports on whether or not a recommendation was helpful. In lieu of this feedback we use this scoring function. Also, note that this method makes the assumption that the user's data is chronologically meaningful. That is to say, we assume that after a user studies a word they will then study words that are related in some way. It is unclear if this assumption is valid. However, note that this assumption will be eliminated with proper user feedback about our recommendations.

3.2 Random Forest Classifier

Now that we have outlined our scoring function, let's turn our attention to the first classifier. Our Random Forest classifier was built to recommend words that a user would be likely to mistake or star given their prior history. It can be found in the `random_forest.ipynb` file of the repository. The model was constructed using `concept_id` and `words_studied` as the features, with `mistaken OR starred` as the label. In other words, this model predicts whether a user is likely to star or mistake a specific word given their current language progress.

After the classifier was constructed, we generated recommendations for each user in the testing data from before. Using the "recommendation" half of each test user's data, we computed the total number of words that the user had studied to that point (the max of the `words_studied` column). We paired this number with the `concept_id` of each word that the user had not yet starred nor mistaken (regardless of whether they had previously studied that word or not).

We fed this data through the forest, and the forest predicted whether or not the user was likely to star or mistake that word. Of those that were predicted to be starred or mistaken, we randomly selected 5 words as our recommendations to the user. Note that introduction of randomness means that our model will not simply predict the same words for each user, thereby avoiding a potential convergence to one set of recommended words instead of a distinct set for each user. In comparing these to the words the user actually studied, we found that this model recommended words with 5.55% accuracy given by our scoring function.

Furthermore, we found that the words recommended were not affected heavily by when a user would be likely to first encounter them. As mentioned previously, words are currently presented within tasks. Users encounter each task at different stages of their progression, so the point at which we expect a a user to see a word for the first time can be approximated based on the tasks in which it appears. Doing another simple query of the back end database we found the average `task_id` of each word. Words with lower average `task_id`s are in easier task and are typically encountered earlier (for example words appearing only in the first task, "Offer a Prayer," have a `task_id` of 6) while words with a higher average `task_id` appear in more difficult tasks and probably encountered later (for instance words that appear only in "Gym" have an average `task_id` of 76). As seen in Figure 5, each `task_id` was about as likely to show up, indicating that the forest did not recommend words simply because they were more common or appeared in easier lessons.

This may seem like a poor model, with only 5.55% accuracy, but we feel that it is quite good given the circumstances. This means about 1 in every 20 recommended words was on target by our standard of accuracy, or 1 out of every 4 recommendation sets contained a relevant word. This is impressive given that we have a pool of nearly 2500 words to choose from, the model is designed to recommend words which are likely to be starred or mistaken (and thus beneficial for the user to specifically study), while the scoring function only recognizes words that were studied in a future point in time. With the added metric of user feedback, we believe that the accuracy of this model will improve. We also believe that our model provides a good enough base line to do some initial beta testing.

3.3 K-Nearest Neighbors Classifier

For our K-Nearest Neighbors classifier, we implemented our own algorithm because of the uniqueness of the dataset (the code can be found in the `KNNEmbark.ipynb` file in the BitBucket repository referenced above). We defined a metric to measure the "closeness" between a user x to predict on and a user b to compare with. This metric is defined as the number of

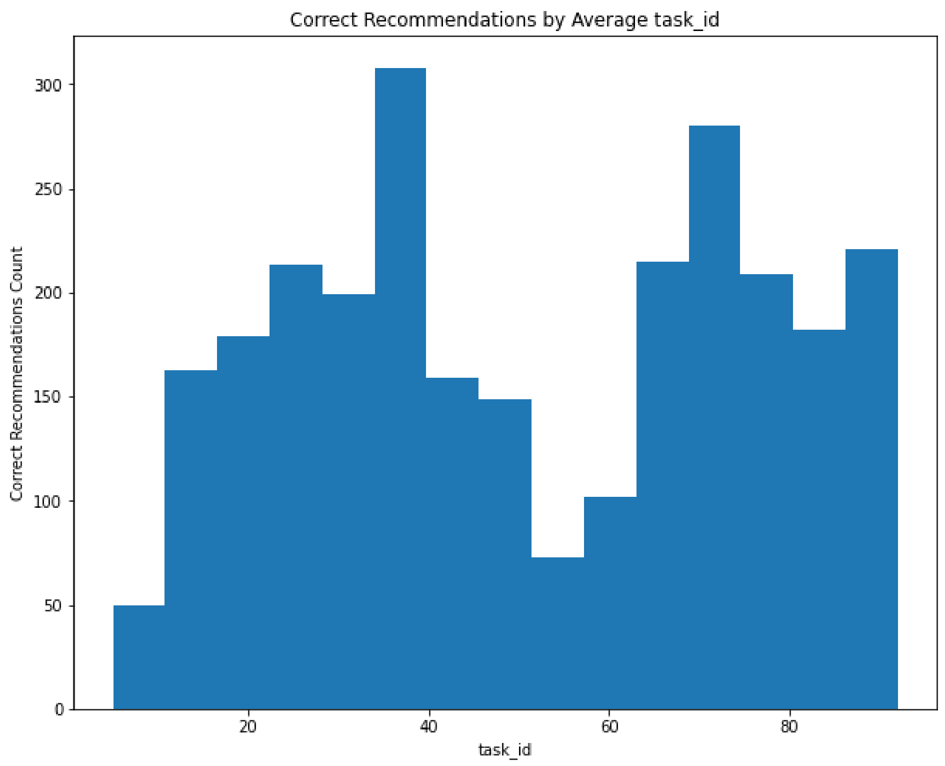


Figure 5: Correct word recommendations by average `task_id`.

words that x has learned that b has not learned. Thus, b is penalized for not knowing words that x knows, but at the same time is not penalized for knowing a word that x does not know. We constructed the metric in this way so that we can identify the most similar user accounts to x while still preserving "future" content, so that we can suggest words that previous users learned next.

We used encoded data as each sample, where each row corresponds to a user and the words they have learned. We then compare a given user x to the data saved in the classifier, and choose the k closest data points, using the metric described above. We then analyze the words that those users later learned, and we select 5 of those words to suggest (with priority given to the words that occur more frequently among those k users).

The performance of this model on test data averaged around 7% to 8% accuracy using the scoring method described above. The model also required around 8 seconds to predict words for all 100 test users, and around 0.1

seconds for 1 test user.

Considering the near 2500 words to possibly suggest, a score of 7% from 5 suggested words is very useful. On average, from each set of 5 suggested words, 1 word was actually starred or mistaken by that user later on.

Although this model is slightly more accurate than the Random Forest classifier according to our scoring function, computational complexity is large concern. For an app serving many language learners, the time to compute these suggested words must remain small. As a result, we disqualified this classifier from further study, since high complexity for prediction is an issue inherent to the KNN algorithm.

4 Ethical Implications

We would be remiss to build these models with no account for the ethical issues of the project. The first ethical consideration is that of the privacy of the user's data that we are employing. When users create an account with churchofjesuschrist.org they are made aware and agree to a privacy notice. As stated in that notice, users agree that their personal data can be collected and that it can be processed for "ecclesiastical, genealogy, humanitarian, social welfare, missionary, teaching, and other operational and administrative purposes." They are made aware that their personal data can be used to "provide ecclesiastical and other related services to fulfill the mission of the Church" and to "customize or personalize features or content on our tools or services." Our purposes are in line with this privacy notice and approved by the developers of this church service.

Second, our goal is to integrate our Random Forest model into the Embark app, but before we do so, it is necessary to consider the potential issues with implementing this feature. It is true that recommendation models have been used in the past, and are still being used today, with the intent to keep the user engaged with an application for as long as possible. These models have no regard for the user's well-being and can foster addiction. The use of such models is ethically questionable. Our model, however, is different from these models in two key ways.

First, our model is not designed to keep a user engaged with the Embark application. It does not take the duration of the user's session into account when recommending words. Second, our model, and the Embark app in general, is not designed to extract value out of the application users. Its sole intent is to aid missionaries in their language learning–there is no ad revenue to be gained by keeping users engaged with the app for a prolonged period. In other words, our model is intended to not harm the users. For these reasons, our model is benign, and we advocate for its use in the Embark app.

5 Impact and Future Action

If the model is implemented in the Embark app there could be profound implications. First, as users interact with the recommendation system, we will get real feedback on the usefulness of our system. As mentioned before, this data will provide us with a true measure of accuracy for our models, and it will obviate the need for our scoring function and test data. Eliminating this accuracy metric also eliminates the assumption that the user data has chronological significance.

This is not all; our model does not exist for its own sake. When our model is implemented within the Embark app it will help missionaries identify the words of most importance to their study. We hope that this will help missionaries know how to focus their language-learning efforts in order to progress most rapidly, which will make them more competent, comfortable, and effective missionaries from their first day in the mission field. We hope also for our model to be applied to other languages, with the only necessary change being a slight modification to our SQL query.

We believe that at this stage, our Random Forest model predicts well enough to be considered for user testing. Although its accuracy is lower than we wished for, we hope that through real-world testing, we can create concrete test cases rather than relying on synthetic test cases. After all, it is more important that a user finds the model useful than whether or not our model can recreate history. We therefore recommend beta testing of our model and believe that it has strong potential to assist missionaries in learning their language.