Intro

Problem Statement

As students at Yeshiva University, we find ourselves immersed in Jewish texts on a daily basis. We have been studying these texts since our earliest years of education, beginning with basic Bible study, the stories of creation and of Abraham, Isaac, and Jacob. As we grow older, we become exposed to many ideas, themes, and concepts of Jewish thought, and we begin to realize that there is so much more that we don't know. We become thirsty, wanting to learn more about things that are still unfamiliar to us, so we turn to the source texts. That's where the problem arises. These source texts can range from esoteric Bible verses, to extensive debates in the Talmud and its commentaries, to works of modern day rabbinic scholars. How can a newcomer to Jewish texts discover the rich source material when they have little familiarity and don't know where to start looking? Until now, students have relied on asking knowledgeable peers and teachers, or consulting a slew of reference books that give some pointers on where to look. This works fine for those with friends who are able and willing to help, or for people in study halls that have these books on their shelves, but for those who don't have access, it can be much more difficult.

A solution to this issue has been proposed by Sefaria, a non-profit organization who has been one of the leaders in online Jewish texts and education over the past decade. Utilizing their library of several hundred million words of Jewish texts, they have begun to develop a catalog of topics by tagging many texts in their library with topics. Anyone curious can search the catalog to find sources from various works of their online library, all in one place.

Until now, Sefaria's topic text taggings have been based on entries in Aspaklaria, an encyclopedia of Jewish thought. They are looking for ways to expand their tagging to the rest of the texts in their online library. To help their efforts, we trained and tested ML models with varying architectures in order to tag more texts. Our basic approach is to make a binary classifier for each topic, where, based on a confidence threshold, the model will label a text as belonging or not belonging to a topic. Overall, we found transformer models to be the most accurate, presenting an accuracy score of at least 92% in our tests.

Our Data

In our project, we had several data sources, both labeled and unlabeled, and we used Data Version Control (DVC) to track changes to and store our data in a Google Drive remote storage destination. All of the data we used was given to us by Sefaria, and consisted of two main files. The first was a file of 400 million Hebrew and English words, collected from Sefaria books, source sheets, and web pages connected to Sefaria citations. Our goal was to use this data to learn word embeddings which we could use to feed into a classification model. The second file was a labeled dataset which contained paragraphs and label pairs, where the paragraphs were taken from various sources in the Sefaria library, and the labels were the topics with which the paragraphs had been tagged.

Sefaria constructed the dataset using Aspaklaria, a Jewish encyclopedia and website which contains a large amount of sources relating to Torah topics. After scraping the text from the Aspaklaria website, Sefaria attempted to match the scraped texts with texts in their own library, and ended up successfully finding more than 100,000 matching paragraphs, tagging each with the Aspaklaria label.

Though this dataset was a good start, it has one main flaw. The goal of our project is to create a multi-label classification model, which requires a multi-label dataset; however, the dataset compiled by Sefaria was not exhaustive. This meant that for any given paragraph, though it might be labeled as belonging to several topics, there could be others it belongs to but hasn't yet been labeled as such. This could pose a problem when training a model on the data, as it could learn (incorrectly) that a given line doesn't belong to a topic when in reality it does. We relied on randomly sampling from unlabeled texts and making an assumption that, if a piece of text was not tagged, it didn't belong to that topic. Ideally, though, we would have a fully specified dataset.

EDA and Cleaning

There were impurities in the Sefaria data which required preprocessing before feeding it to our models. Firstly, the texts from both the labeled and unlabeled data contained english characters and words. Also, both datasets included many marks of punctuation. For the Hebrew text in particular, abbreviations containing single quotation marks (called *roshei teivot*) appear quite frequently in rabbinning texts, including the talmud, its commentaries, and the like. Many of these abbreviations appear quite frequently in the texts, and they don't provide any substantive meaning, and act more like glue that connects clauses together.

For dealing with English characters and words, we had two strategies: 1) completely removing any line which contained English from our dataset, or 2) just removing the English from but leaving the rest of the line (provided it also contained Hebrew characters) in the dataset. For *roshei teivot*, we had three strategies: 1) leaving the *roshei teivot* in the data as is, and perhaps the model will learn the their meaning like any other word; 2) removing the quotation marks from the *roshei teivot*; and 3) treating them like stopwords and removing *roshei teivot* altogether from the dataset.

Modeling

Binary Classification

To solve the multi-classification problem, we chose to create binary classification models for each topic, giving labels of 0 (text doesn't belong to the topic) or 1 (text does belong to the topic). We began by creating models for the top five most common topics in the dataset.

Preparing the Datasets

For each topic, we subsampled from the original dataset, forming a smaller set in which half of the texts belonged to the training topic (labeled as 1), and half didn't not. The records in the second (negative) half of the set were chosen at random from texts that weren't labeled with the topic label. Finally, we performed a train/test split on the data. To represent the data numerically in order to pass it to the models, we created word bags from the training data using sklearn's CountVectorizer, removing stopwords (e.g. 'a', 'and', 'the) in the process. We also tried using the TfidfVectorizer, though results didn't seem to differ significantly.

Another common approach to representing text is using already learned word embeddings, where each word has a vector representation. To represent a sentence, one can take the average of all the word embedding, and use the newly formed vector to pass into a model. We took 2 approaches: 1) we trained our own word2vec model on 500,000 lines of text from all_text_he.txt, and 2) we used a pre-trained model from Dicta which was trained on rabbinic literature. After running the datasets through a basic logistic regression model, the accuracy results for both word2vec approaches failed to meet the accuracy reached using word bags.

Overview of Sklearn and XGBoost Models

The following are brief explanations of the modeling approaches that we used. Before training each model, we prepared the data as explained in the previous section. Model performance metrics are shown below.

Naive Bayes

The approach of this statistics-based model is rooted in Bayes' Theorem and is implemented by Sklearn in their MultinomialNB class. The model estimates the priors and conditional probability distributions of the features in the training set. During inference, these estimations are then used to predict which class a new data sample is most likely to belong to. For each class, the model adds the prior to the product of the conditional probabilities of each feature, and the class with the highest result wins.

Logistic Regression

Logistic regression is a common modeling choice for binary classification problems and is implemented by Sklearn in their LogisticRegression class. The goal in training a logistic regression is to learn the most optimal combination of weights to apply to the input features that will predict the ground truth label most accurately. These weights are learned through an iterative process using the gradient descent algorithm. After each pass through the data, the error (calculated by a loss function) is determined, and the goal is to learn to adjust the weights after each iteration in the way that decreases the loss the most. At the end of the model, the results are passed through the sigmoid function, which compresses the output values to a range between 0 and 1. In general, if the result is greater than .5, it means the input data belongs to the given class, otherwise it doesn't.

K Nearest Neighbors (KNN)

This model is Sklearn's implementation of the KNN algorithm and is implemented in the KNeighborsClassifier class. No parameters are learned during training. During inference, the model compares the input to all of the training samples, finding the top *k* samples that are closest to the inputted data using a distance metric (commonly Euclidean distance). The prediction is determined based on the class who has the most number of samples in the *k* closest samples to the input data.

Decision Trees

Sklearn provides an implementation in its DecisionTreeClassifier class. During training, the data is split using the feature that provides the highest amount of information gain, determined by metrics such as entropy or Gini impurity. After the split, the process of splitting by the most useful feature is repeated recursively on each subset of data. The algorithm terminates when the depth threshold has been reached or the data can no longer be split. For inference, the input sample is run through the different splits of data until it reaches a leaf node at the bottom of the tree. The class of the leaf node determines the label of the input sample.

Random Forest

The random forest classification algorithm is an ensemble method which is implemented by Sklearn in the RandomForestClassifier class. During training, the algorithm takes many random subsamples of data, each subsample only keeping a randomly selected subset of the original dataset's features, and trains a decision tree on the subsampled data. During inference, the input sample is run through each of the trees, and the most commonly outputted class determines the class of the input sample.

XGBoost

This algorithm is another type of tree ensemble which is implemented by XGBoost in the XGBoostClassifier class. During training, the algorithm builds a series of decision trees, each one learning from and trying to improve on the mistakes of the previous tree. For inference, the input sample is passed to each tree, and the prediction is determined by a weighted sum of the predictions from each tree.

Transformers

After trying out the basic models offered by Sklearn and XGBoost, we seemed to hit a ceiling on accuracy (averaged across topics - around 85%), which represented a more fundamental issue - the models didn't seem to be learning a contextual understanding of the data. This makes sense, since the models were using a bag-of-words representation, which is context agnostic. Therefore, it became clear that to get a boost in metrics, we should pursue transformer models.

We chose to take the following approach: find pre-trained transformer large language models (LLMs), particularly ones that were trained on the Hebrew language, then fine-tune binary-classification models on a per-topic basis. Attention-based LLMs are the state-of-the-art

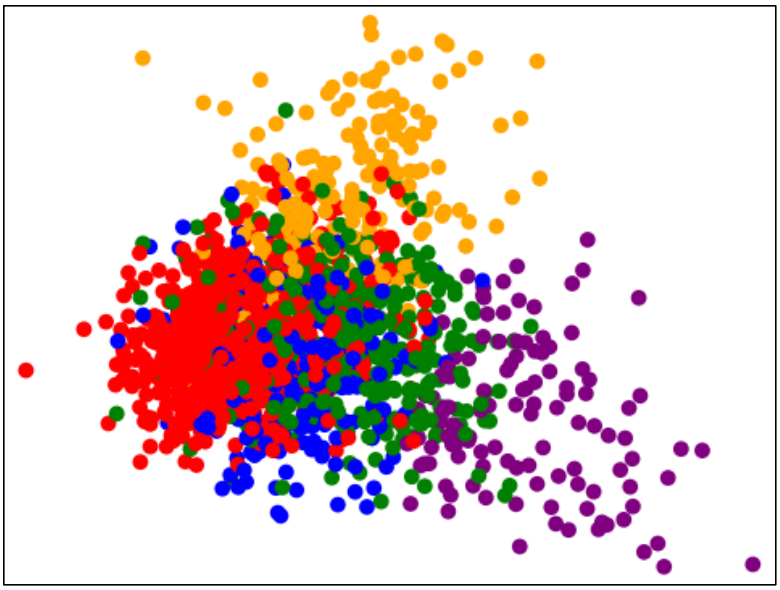
in natural language processing (NLP) for understanding semantics and context in text, so they seemed to be the best solution. Our first decision was whether to go with an English or Hebrew model. On the one hand, English models have been trained on a lot more data, and therefore have better general-language semantics understanding, but tuning them for our use case would require more experimentation as to where to cut off the model and retrain on Hebrew language data. Therefore, we chose to go with the Hebrew models that were a more organic fit for our problem space and would take less time to train (since fewer new layers are required), especially since our solution requires a new binary-classification model for each topic.
There are 3 main BERT-like models for the Hebrew language, all of which we used:

- AlephBERT is presented as a good general-purpose model for getting started on Hebrew NLP projects. The main benefit was that it is naturally loaded as a classifier model, which is fit for our use case.
- HeBERT is a more particularistic model, trained to do sentiment analysis, which is a natural fit for our use case as well (polarity is similar enough to binary-classification of IS TOPIC / IS NOT TOPIC).
- BEREL - as opposed to the previous models, this was trained on Torah texts, which should be a better fit for our data which includes Biblical, Mishnaic, Talmudic, and Rabbinic Hebrew. On the other hand, it was trained as a [MASK] fill-in model, which is not the best fit for our classification use case. (Upon application, the results showed that this had no effect, and BEREL performed the best out of all three transformer models, as shown below)

Other Modeling Strategies

Expectation Maximization (EM)

Due to our large amount of unlabeled data, and our problem being of of multi-classification, we thought we should try a method of semi-supervised, soft-classification learning using the Sklearn's GaussianMixtureModel, an implementation of the Expectation Maximization (EM) algorithm. In our trial run, we limited the data to the five most common topics. The results weren't great, with the model reaching 30% accuracy on predicting which class that the text most likely belonged to. Once again, we represented the text using bag-of-words, TF-IDF, and pre-trained word embedding vectors, with the results not varying greatly. After training the model, a visualization of the results showed that the model did reasonably well in clustering each topic together and not leaving data points scattered in random places; the problem, though, was that the clusters overlapped one another heavily, making it difficult to distinguish them from each other. Below is a sample visualization:



Visualization of the EM model results. Some clusters are relatively well defined, though most are overlapping

Results

Below are the test results after training our most effective model architectures on the top five most common topics: לימוד (Limud), תורה (Torah), תשובה (Teshuva), תפילה (Tefillah), and ישראל (Yisrael):

לימוד (Limud) model	**Accuracy**	**Precision 0**	**Recall 0**	**F1-score 0**	**Precision 1**	**Recall 1**	**F1-score 1**
alephBERT	0.86	0.83	0.88	0.85	0.89	0.85	0.87
BEREL	0.92	0.9	0.93	0.91	0.94	0.91	0.92
heBERT	0.88	0.86	0.88	0.87	0.89	0.87	0.88
Multinomial NB	0.82	0.74	0.86	0.8	0.88	0.78	0.83
Logistic Regression	0.79	0.85	0.75	0.8	0.73	0.83	0.78
K Neighbors Classifier	0.62	0.96	0.56	0.71	0.28	0.89	0.43
Decision Tree Classifier	0.78	0.78	0.77	0.77	0.77	0.79	0.78

Random Forest Classifier	0.8	0.79	0.81	0.8	0.82	0.8	0.81
XGB Classifier	0.79	0.81	0.78	0.79	0.77	0.81	0.79

For Limud, the BEREL model performs the best in almost every metric and usually by a large margin. Even the other two transformer models failed to come close to the BEREL model. Also, none of the SKlearn models nor the XGB model performed particularly well in this topic. Overall, the Limud topic was the worst performing of all of the models.

תורה (Torah) model	**Accuracy**	**Precision 0**	**Recall 0**	**F1-score 0**	**Precision 1**	**Recall 1**	**F1-score 1**
alephBERT	0.94	0.92	0.95	0.94	0.95	0.92	0.94
BEREL	0.94	0.93	0.94	0.94	0.94	0.93	0.94
heBERT	0.9	0.89	0.91	0.9	0.92	0.89	0.9
Multinomial NB	0.79	0.71	0.85	0.77	0.88	0.75	0.81
Logistic Regression	0.86	0.93	0.82	0.87	0.8	0.92	0.85
K Neighbors Classifier	0.66	0.96	0.6	0.74	0.36	0.89	0.51
Decision Tree Classifier	0.88	0.89	0.87	0.88	0.86	0.88	0.87
Random Forest Classifier	0.89	0.86	0.91	0.88	0.91	0.87	0.89
XGB Classifier	0.9	0.91	0.89	0.9	0.89	0.9	0.89

For Torah, all models performed fairly well, with alephBERT and BEREL performing the best. All models, except for K Neighbors and Multinomial NB, had scores in the 0.80s and 0.90s. The XGB model here performed better than the SKlearn models, but not by much, as the f1-scores between the XGB and Logistic Regression, Random Forest Classifier, and Decision Tree Classifier are all between 0.05.

תשובה (Teshuva) model	Accuracy	Precision 0	Recall 0	F1-score 0	Precision 1	Recall 1	F1-score 1
alephBERT	0.9	0.88	0.93	0.9	0.92	0.86	0.89
BEREL	0.95	0.98	0.94	0.96	0.92	0.98	0.95
heBERT	0.92	0.94	0.92	0.93	0.91	0.92	0.92
Multinomial NB	0.81	0.73	0.9	0.8	0.9	0.74	0.81
Logistic Regression	0.89	0.97	0.85	0.9	0.79	0.95	0.87
K Neighbors Classifier	0.63	0.99	0.6	0.74	0.21	0.93	0.35
Decision Tree Classifier	0.88	0.9	0.87	0.89	0.85	0.88	0.86
Random Forest Classifier	0.88	0.93	0.87	0.9	0.83	0.91	0.87
XGB Classifier	0.89	0.95	0.86	0.9	0.82	0.93	0.87

For Teshuva, the transformer models again perform better than the SKlearn and XGB models. Here, the BEREL model performed exceptionally well, with accuracy and both f1-scores being 0.95 or better. Also, recall 1 is 0.98, which is the best out of all the models. For almost all models, their best performance was on the Teshuva topic.

תפילה (Tefillah) model	Accuracy	Precision 0	Recall 0	F1-score 0	Precision 1	Recall 1	F1-score 1
alephBERT	0.89	0.91	0.88	0.89	0.87	0.9	0.88
BEREL	0.93	0.94	0.94	0.94	0.93	0.93	0.93
heBERT	0.91	0.91	0.91	0.91	0.91	0.91	0.91
Multinomial NB	0.84	0.79	0.88	0.84	0.89	0.8	0.84
Logistic Regression	0.84	0.92	0.81	0.86	0.77	0.9	0.83
K Neighbors Classifier	0.63	0.95	0.59	0.73	0.28	0.85	0.42

Decision Tree Classifier	0.85	0.9	0.83	0.86	0.81	0.88	0.84
Random Forest Classifier	0.88	0.95	0.83	0.89	0.8	0.94	0.86
XGB Classifier	0.84	0.92	0.79	0.85	0.74	0.9	0.81

Here as well, the BEREL model performed the best with some of the highest scores. It has the best accuracy and f1-scores compared to the rest of the models. Here it is interesting to note how the Random Forest Classifier performed the best out of all of the non-transformer models.

ישראל (Yisrael) model	**Accuracy**	**Precision 0**	**Recall 0**	**F1-score 0**	**Precision 1**	**Recall 1**	**F1-score 1**
alephBERT	0.88	0.86	0.92	0.89	0.92	0.85	0.88
BEREL	0.93	0.9	0.96	0.93	0.95	0.9	0.93
heBERT	0.88	0.84	0.92	0.88	0.92	0.84	0.88
Multinomial NB	0.76	0.63	0.87	0.73	0.89	0.68	0.77
Logistic Regression	0.84	0.88	0.83	0.85	0.8	0.85	0.83
K Neighbors Classifier	0.73	0.88	0.7	0.78	0.56	0.81	0.67
Decision Tree Classifier	0.79	0.78	0.82	0.8	0.81	0.76	0.79
Random Forest Classifier	0.85	0.82	0.89	0.85	0.89	0.81	0.85
XGB Classifier	0.84	0.84	0.86	0.85	0.85	0.82	0.83

For Yisrael, the BEREL model performs the best once again, this time in every metric. The non-transformer models have f1-scores of only 0.85 at best. Interestingly, the K Neighbors Classifier performed significantly better on this topic than on any other test topics.

Overall, the BEREL transformer models performed the best, beating all other models on all five of our test topics. The K Neighbors Classifier performed the worst in all models. The rest of the models all performed similarly.

Model Productionization

For productionization of our models, we created a simple website that takes two inputs: text, which is passed to our models to be tagged and a confidence threshold, used after running inference with each model. The input text first gets cleaned using the same methods we used to clean our data. Then, it is passed into the logistic regression and all three transformer models. The models return their predictions and confidence probabilities, all of which get displayed as on the site. The final prediction is then determined by the threshold - if the confidence level is above the threshold, it is classified as belonging to that topic.

Further improvements

Multi-headed Model

After training a separate transformer model for each topic, we realized that space and inference turnaround was an issue - each model took up about one Gigabyte of space, and took about one second to return a single inference. This means that the architecture will not scale for for a large number of models, since each model entails: onboarding, training time of a new fine-tuned transformer; additional inference runtime, because running the text through another large model requires even more time; and more space, each model will use an addition GB of storage.

The solution to at least the latter two problems would be an architecture shift, where only one base pre-trained model is saved, and we attach many fine-tuned heads (trained on top of the base model) to the end of the model for each topic. This would save space (since only one full LLM is stored), and a big latency boost (since only one pass through the LLM is required, after which the single interim output can be forward-fed through the heads to get the classifications).

Analysis of Results

It would be interesting to see how accurate the models are for each type of text (e.g. Bible, Talmud, works of Maimonides, etc). The dialects of Hebrew in Sefaria's library include Biblical, Rabbinic, and modern, and there is a large amount of Aramaic. It's possible that models fine-tuned on the different dialects and literature types would perform better than our one-size-fits-all model.

Construct a Better Dataset

Creating a perfect training data for a multi-classification problem wasn't really feasible for us, since we did not have negatively labeled data (i.e. the text was labeled as NOT TOPIC X, rather only IS TOPIC Y), but we got around this issue by creating training data for binary-classification (negative labels were artificially created by taking random samples from

texts not labeled as positively the given topic). Ideally, we would have liked to be able to train a classifier on a fully specified, exhaustive dataset, where each paragraph of text had a complete list of topics related to it, which would imply that it truly doesn't belong to any topic not listed.

Conclusion

In sum, we implemented a ML solution to Sefaria's text tagging problem. Using the training data that Sefaria provided to us, we tested various modeling techniques and produced some promising results. We achieved the best results using the state-of-the-art transformer models, which are focused on understanding the broader context of a piece of text, with accuracy scores in the mid 90s. We hope that our models will be useful to Sefaria and aid them in their goal to provide easy access to new material for those who are looking to gain more knowledge in topics based within Jewish texts.