Stanford Recruiting Emails
Free Food Prediction and Subject Line Generation

Leo Mehr
leomehr@stanford.edu

Abstract

This project presents a novel dataset scraped from the Stanford Computer Forum recruiting email list archives, followed by two prediction tasks: one traditional ML classification task and one difficult generative language task. Our classifiers yielded accuracies of nearly 90% on the food prediction task and our generative language models produced realistic subject lines but yielded F1 scores of approximately 10%.

1 Introduction

Recruiting is exceedingly important at Stanford. Companies come looking for top talent, students seek the perfect job, and Stanford wants its students to thrive beyond graduation. The Stanford Computer Forum Forum is a Stanford-run organization that was founded over 50 years ago and is the primary force for bridging the gap between industry and Stanford's student body. The Forum runs an email list titled "recruiting" List, by which they send emails to current students (BS, MS, and PhD) for various recruiting events and opportunities.

This project has three main goals: (1) produce an ML-quality dataset of Stanford's CS recruiting emails by scaping, parsing, and cleaning the raw source text; (2) predict whether an email describes an event with food based solely on the subject line; (3) generate an accurate subject line given an email body.

In the following sections, we present the recruiting dataset (sec 2), tasks performed (sec 3), the ML methods used (sec 4), results from experiments (sec 5), and conclude the project (sec 6).

Related Work The most well-known email dataset comes from the Klimt and Yang [2004] paper "The Enron Corpus: A New Dataset for Email Classification Research." The authors presented techniques for classifying emails into user-specific folders using SVMs over bag-of-words representations of emails. The Enron email corpus has since been used for a variety of NLP, ML, and network analysis research (the original paper has over 800 citations). Although generally there has been little research on subject line generation for emails, one paper by Dredze et al. [2008] generated summary keywords from email bodies in the Enron dataset. A very closely related task to subject line generation is text summarization, specifically the task of news headline generation. Banko et al. [2000] presented a generative method inspired by statistical translation research, in which they model the likelihood of a headline and produce a sequence that is an approximate maximum likelihood estimate.

2 Dataset

We used the mailman [1] interface to find the archive of every email sent to the recruiting email list [2]. The archive is stored on a fileserver containing one file per month with every email of that month concatenated together in a specific format [3]. The archive contains every email sent to the recruiting list since April 2007, which totals to over 12 years of emails.

Once scraped, the entire corpus of recruiting emails is 16MB of text. We wrote a script that merges all the raw data and parses out the email subject and body texts, producing 6123 emails containing 463,895 lines in total. 45 of these emails contained malformed headers and were discarded.

3 Tasks

The two tasks presented are inspired by two different stakeholders of the recruiting email list: the students and the Stanford Computer Forum.

1. `has_food` – Predict whether or not food will be provided at an event, solely based on the Subject line of the email. It is very well known that students often look for free food at events, and further that students do not want to bother reading whole emails. This task presents the challenge of first producing a supervision signal, since the dataset is unlabeled (discussed below), and then effectively modeling the short, declarative text of email subjects.
2. `gen_subject` – Predict subject line based on the text of the body. The Stanford Computer Forum could benefit from an algorithm that can automatically generate titles for events based on the email body. At the very least, such an algorithm could act as a recommendation engine to propose titles that are more consistent or well-suited for Stanford recruiting events.

3.1 Formalizing the tasks

The dataset is a collection of emails $E = \{e^{(1)}, e^{(2)}, ..., e^{(n)}\}$, where each $e^{(i)}$ is a single email sent to the recruiting list, and is composed of a subject $s^{(i)}$ and a body $b^{(i)}$. We use subscripts to denote the tokens in a piece of text, so for example $s_j^{(i)}$ refers to the j-th word and $b^{(i)}$ can be used interchangeably with $b_1^{(i)}, b_2^{(i)}, ..., b_{-1}^{(i)}$, where $b_{-1}^{(i)}$ indicates the last token of $b^{(i)}$.

`has_food` is a classifcation task in which we are given solely $s^{(i)}$ and try to predict label $f^{(i)} \in \{0, 1\}$, which indicates whether or not email $e^{(i)}$ describes an event that provides free food. `gen_subject` is a generative task in which we are given $b^{(i)}$ and attempt to produce a $\tilde{s}^{(i)}$ that is linguistically equivalent to $s^{(i)}$.

3.2 Evaluation Metrics

In `has_food` , we specifically seek to have a model with high accuracy and precision. While accuracy is standard for classification, we specifically also choose precision because we aim to have a low false-positive rate. That is, we want to avoid predicting that an event has food when it does not, since telling a student there is free food when there is none will surely lead to hunger, anger, and frustration. While we also care to avoid false negatives, missing food events is understandably less disasterous.

In `gen_subject` , linguistical equivalency is challenging to evaluate. To do so accurately, we would need a human to observe the output and determine whether or not $\tilde{s}^{(i)} \approx s^{(i)}$. However, it is immediately clear that such an evaluation technique does not work – human evaluation is expensive, slow, and unscalable. In the field of Machine Translation, which also requires similar evaluation of generated text, a popular approach to use automated metrics such as BLEU [Papineni et al., 2002]. For this task, however, unlike in Machine Translation, we at least have one true target sequence $s^{(i)}$ (even though others could be valid). Thus, we propose to use the metrics of Exact Match (EM) and F1, exactly as proposed for evaluating Question Answering systems [Rajpurkar et al., 2016]. EM is

[1] Stanford's mailing list interface `https://uit.stanford.edu/service/mailinglists/tools`

[2] `recruiting@lists.stanford.edu`

[3] E.g. see `https://mailman.stanford.edu/mailman/suprivate/recruiting/2019-May.txt`

defined as $1/n \sum_i^n 1\{s^{(i)} = \tilde{s}^{(i)}\}$. To calculate F1, we use bag of words overlap between $s^{(i)}$ and $\tilde{s}^{(i)}$ to produce precision and recall scores.

3.3 Deriving supervision signal for `has_food`

As described above, `has_food` is a supervised task, but the original data does not contain the labels $f^{(i)}$. Instead, the labels must be inferred from the email bodies, in the same way that a student would read an email to determine whether it mentions food. The insight here is that inferring $f^{(i)}$ from $b^{(i)}$ is actually quite easy, while inferring from $s^{(i)}$ is harder. We wrote a labeling function (i.e. interface $b^{(i)} \rightarrow$ `boolean`) that checks whether the body contains any one substring from a set of phrases [4]. In order to evaluate the efficacy of our labeling function, we randomly sampled 100 emails from the dataset and manually classified them, then compared our labeling to that of our labeling function. We found that our labeling function performs extremely well, with an accuracy of 97%, precision 98%, and positive recall 96%. Further, we find that the dataset is very well balanced, with 51.3% of all emails containing food.

4 Methods

4.1 `has_food`

We use two popular approaches: (1) `NBC` a Naive Bayes classifier with Laplace smoothing; (2) `SVM` a Support Vector Machine over bag-of-words features. [5]

`NBC` is a generative classification algorithm that models $p(s^{(i)} \mid f^{(i)})$, the probability of a subject line given whether there is free food, and $p(f^{(i)})$, the probability of an email describing an event with free food. By making the Naive Bayes assumption that every token in the subject is independent, we make the approximation:

$$p(s^{(i)} \mid f^{(i)}) \approx \prod_{j=1}^{\texttt{len}(s^{(i)})} p(s_j^{(i)} \mid f^{(i)})$$

`SVM` is a discriminative classification algorithm that instead models $p(f^{(i)} \mid s^{(i)})$ directly. In order to discretize our subject text $s^{(i)}$ into a numerical feature vector, we treat it as a bag of words and create a vector of length $|Vocab|$ with 1s in the position of any contained words. Using a soft-margin SVM (we certainly do not expect this data to be linearly separable), we know that this is equivalent to the emperical risk minimization of hinge loss with l2 regularization. Using stochastic gradient descent, the loss function for a single featurized example $s^{(i)}$ is:

$$\ell(\theta) = \max\left(0, 1 - f^{(i)}(\theta^T s^{(i)})\right) + \lambda\|\theta\|^2$$

4.2 `gen_subject`

Recall that the task is to predict $\tilde{s}^{(i)}$ given $b^{(i)}$. We implement 3 different language models [6] from scratch: (1) `subject-bigram` creates a bigram language model using solely subject lines and ignoring the body text; (2) `body-unigram` creates a unigram language model and incorporates body text; (3) `body-bigram` creates a bigram language model and combines information from subjects and bodies.

In each of these three language models, we model the probability of generating the $j+1$th token of the subject line:

$$p(s_{j+1}^{(i)} \mid s_1^{(i)}, ..., s_j^{(i)}, b^{(i)})$$

In order to generate a subject line $\tilde{s}^{(i)}$ for a given body, we begin with a special start token $s_0^{(i)} =$ `START` and iteratively sample from the conditional distribution to produce every $s_j^{(i)}$ until we generate

[4] 'food', 'dinner', 'lunch', 'breakfast', 'will be served', 'will be provided', 'pizza', 'boba', 'sushi'

[5] We take advantage of `scikit-learn`'s implementation of `MultinomialNB` (Multinomial Naive Bayes) and `SGDClassifier` (Stochastic Gradient Descent) classifiers Pedregosa et al. [2011].

[6] a language model is roughly defined as a probabilty distribution over sequences of words

Table 1: `has_food` performance

Model	Split	Accuracy	Precision
NBC	dev	0.893	0.905
	test	**0.878**	0.876
SVM	dev	0.902	0.904
	test	0.861	**0.885**

Table 2: `gen_subject` performance

Model	Split	F1	Precision
subject-bigram	dev	0.097	0.108
	test	0.112	0.124
body-unigram	dev	0.085	0.111
	test	0.107	0.145
body-bigram	dev	0.102	0.159
	test	**0.118**	**0.184**

the special `END` token or `MAX_LEN` (default 30). However, each of our 3 approaches approximates the above conditional distribution differently.

(1) `subject-bigram` ignores the body text:

$$p(s_{j+1}^{(i)} \mid s_j^{(i)})$$

(2) `body-unigram` only uses body text and ignores previous generated tokens:

$$p(s_{j+1}^{(i)} \mid b^{(i)}) \approx \prod_{k=1}^{\texttt{len}(b^{(i)})} p(s_{j+1}^{(i)} \mid b_k^{(i)})$$

(3) `body-bigram` combines the above two approaches:

$$p(s_{j+1}^{(i)} \mid s_j^{(i)}) \cdot p(s_{j+1}^{(i)} \mid b^{(i)}) \approx p(s_{j+1}^{(i)} \mid s_j^{(i)}) \cdot \prod_{k=1}^{\texttt{len}(b^{(i)})} p(s_{j+1}^{(i)} \mid b_k^{(i)})$$

5 Results

Before running any experiments, we split the original dataset into train, dev, and test sets. The split was approximately 80-10-10, with the test set containing every email from 2015 and from May 2019. Throughout development iterations, only the train and dev splits were used. The test split was (with some perseverence and resolve) used only once at the very end of model development when all the models were finalized, and did not influence any changes to the models.

5.1 `has_food`

Table 1 illustrates the performance of our two models – the Naive Bayes Classifier `NBC` and Support Vector Machine `SVM` – on the dev and test sets. Overall, we are pleased with the performance of our two models on the tasks, with an overall test accuracy of 87.8%. Both models perform similarly well, with `SVM` yielding better precision and `NBC` yielding better accuracy. Yet we observe a small but consistent degredation from the dev to the test set, which we believe are explained by two reasons: while developing our models, we overfitted our models and their parameters to specifically yield good performance on the dev set, and that the test set is not quite representative of the overall dataset.

`NBC` chooses the class that maximizes the product of word probabilities conditioned on class (and the probability of the class), so it needs $2 * |Vocab| + 2$ parameters in total, while `SVM` needs only $|Vocab|$. Because the vocabulary size of subject lines is reasonably small, 2477, this is not an issue, but it is clear that in some scenarios the generative classification model `NBC` would be more expensive in space complexity.

Tables 3 and 4 present feature analysis of our two classification models. These features are telling of how the models work and point out some very interesting differences. In `NBC`, we observe that the feature scores are all negative – this is because they represent the log probabilities. We present the 6 features from each class with the largest log probabilities, and observe that there is overlap of basic non-informative words such as "reminder" and "in" that are common in both food and non-food emails. In order to remove the importance of such words, we implemented TF-IDF instead of using basic word counts, but found that this did not boost the performance of `NBC`. In `SVM`, however, the

Table 3: `has_food` NBC features

No Food	Score	Score	Has Food
the	-3.85	-3.27	reminder
for	-3.88	-3.19	104
reminder	-3.94	-3.07	info
in	-3.99	-3.04	gates
digest	-4.04	-3.04	session
of	-4.16	-3.02	in

Table 4: `has_food` SVM features

No Food	Score	Score	Has Food
from	-1.154	1.335	joblunch
drop	-0.938	1.046	coming
group	-0.902	0.974	company
counseling	-0.793	0.866	lunch
fairs	-0.757	0.866	dinner
oracle	-0.757	0.829	splunk

feature scores range from negative to positive values, because there is only a single feature vector θ that is used to make a prediction $\text{sign}(\theta^T s^{(i)})$. Here, we do not observe common words, since they will not be strong predictors of whether an event has food. Instead, we see words that are highly predictive of either class, such a the very words "lunch" and "dinner" for food events, and "counseling" or "fairs" (as in Career Fairs) for non-food events. It is amusing to see that companies surfaced in the results, with "Oracle" for no food and "Splunk" for food.

5.2 `gen_subject`

Table 2 illustrates the performance of our 3 generative language models. We computed Exact Match for all entries as well, but found that it was always 0 – none of the subject lines generated by our models matched the actual subject lines. Despite no exact matches, we see that the F1 and Precision scores range from 10 to 20%. That is, on average part of the subject lines are reconstructed, but never perfectly reconstructed. It is clear that EM is a very difficult metric for this task, since even human performance would likely yield very low EM (the subject lines are long and slight variations in language make them very difficult to predict exactly). Nonetheless, out models still have low F1 and Precision scores, and are likely due to the appearance of basic common words such as "Reminder", "Digest", "Info", etc. One unexpected result from the experiments was that test performance was generally better than dev. After looking more closely at the test set, we believe that a larger fraction of emails in 2015 contained common, consistent prefixes such as "SAVE THE DATE" and "FINAL REMINDER", tokens that our models often generate. Several examples of the `body-bigram` model include "FINAL REMINDER Computer Forum", "Reminder Uber Info Session 30pm in Gates 104", and "REMINDER RSVP by March". While the isolated subject lines were relatively convincing on their own, they are often competely wrong about the company and details of the event, since they fail to isolate the most salient facts from the email body – difficult for statistical approximations but easy for humans.

6 Conclusion

We produced a novel dataset of recruiting emails and solved two interesting tasks: one that required automatic generation of supervision labels and another that proved to be a very difficult generative language task. While `gen_subject` yielded weak results, our models performed very well on `has_food` – these results are illustrative of the trends that generating language is hard (the entire field of statistical machine translation as an example), while traditional prediction tasks are solveable with classic ML approaches and simple bag-of-words assumptions.

Future work One very important direction for future work would be to use neural networks for both predicition tasks. LSTM networks with attention are able to capture long-range dependencies in text and have had huge success in modern NLP research. The sequence-to-sequence architecture has proven very successful for summarization tasks Chopra et al. [2016] and for news headline generation Tan et al. [2017]. Pre-trained word embeddings such as ELMO McCann et al. [2017] have also boosted the performance of many NLP tasks. One different direction of future work would be to add supplemental data such as event attendance counts, or counts of how many students read vs. deleted recruiting emails.

We would like to thank professors Chris Re and Tengyu Ma for teaching this excellent iteration of CS 229, and especially head TA Anand Avati for very insightful help during project office hours.

References

Michele Banko, Vibhu O Mittal, and Michael J Witbrock. Headline generation based on statistical translation. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, pages 318–325. Association for Computational Linguistics, 2000.

Sumit Chopra, Michael Auli, and Alexander M Rush. Abstractive sentence summarization with attentive recurrent neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 93–98, 2016.

Mark Dredze, Hanna M Wallach, Danny Puller, and Fernando Pereira. Generating summary keywords for emails using topics. In *Proceedings of the 13th international conference on Intelligent user interfaces*, pages 199–206. ACM, 2008.

Stanford Computer Forum. `https://forum.stanford.edu/`.

Bryan Klimt and Yiming Yang. The enron corpus: A new dataset for email classification research. In *European Conference on Machine Learning*, pages 217–226. Springer, 2004.

Stanford "Recruiting" Email List. `https://mailman.stanford.edu/mailman/listinfo/recruiting`.

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. Learned in translation: Contextualized word vectors. In *Advances in Neural Information Processing Systems*, pages 6294–6305, 2017.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics, 2002.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, 2016.

Jiwei Tan, Xiaojun Wan, and Jianguo Xiao. From neural sentence summarization to headline generation: A coarse-to-fine approach. In *IJCAI*, pages 4109–4115, 2017.