Analysis of Structure in Resumes for Prediction and Classification

Sophia X Cui

W 266: Natural Language Processing
UC Berkeley School of Information
sophia@ischool.berkeley.edu

Abstract

Semi-structured documents encompass a wide corpus of documents available on the web, e.g. medical records, online profiles, semantic linked data, Wikipedia. For these types of documents, metadata embedded in the structure as well as the hierarchy often provides additional insight for contextual understanding and interpretation. While a larger bold font denotes a section heading to human readers, machine learning may not glean the importance of the code that styled such text. This project explored the impact of structure and metadata from a set of semi-structured HTML-formatted resumes with respect to classification and prediction. The relatively high accuracy rates obtained by bidirectional LSTM RNNs on several types of predictions suggest that utilizing the structure and metadata of documents could be a promising approach to machine learning on such corpi.

Keywords*: Semi-Structured Documents; Resumes; Bidirectional LSTM RNN*

1. Introduction

"Semi-structured data is a form of structured data that does not conform with the formal structure of data models associated with relational databases or other forms of data tables, but nonetheless contains tags or other markers to separate semantic elements and enforce hierarchies of records and fields within the data." - Wikipedia

There has been rapid growth of digital semi-structured data in the last couple of decades, examples being medical records, online profiles and linked semantic data. Many technologies and tools also gained prominence as mediums of exchange, such as json or XML for loose structure definition and MongoDB and NoSQL for storage. Parsing, extraction and prediction on these types of data set has also spurned different kinds of tools. From MongoDB query language to JQuery class selectors, loosely parsing and querying this type of information is a different paradigm from both traditional relational databases and plain text.

While machine learning based NLP has primarily focused on raw text as the primary input, there has been increasingly more research on how deep learning can be applied to semi-structured data for information extraction and prediction [1][2][3]. Neural language models such as bidirectional LSTM RNNs can capture remote dependencies across forward and backward time scales which suits structured text embedded with open and end tags. This project explores the impact of HTML structure in semi-structured resumes for prediction in bidirectional LSTM RNNs using TFIDF logistic regression as baselines.

2. Project Overview

2.1 Background

Previous work successfully leveraged HTML markup structure in web documents with deep neural networks to successfully parse and extract information [4][5]. Other work extracted structure and clustered text as a means to narrow the search space and produce more accurate results [6][7]. Meanwhile, there's a growing multitude of online content served over dynamic pages: fixed style and structure, dynamic content. Many such content present opportunities for data-mining and NLP based machine learning, but it's unclear how much the HTML markup on these pages could help current neural models improve upon plain text classification or prediction.

This project aims to deep dive and explore the impact of consistently decorated HTML for classification and prediction on a domain specific dataset of resumes.

2.2 Datasets Utilized

We used a scraped dump of 8 million unique English resumes from Indeed[1] from 2017. The resumes are in HTML with consistent classes and ids for corresponding sections of the resume. For example, individual work experiences are classed 'work-experience-section' which nests under a work experience section, with id 'work-experience-items'.

We also used Word2vec Slim[2] as pretrained embeddings, with a vocabulary size of 300k trained from Google News. Because the vocabulary of resumes is dominated by common English terms, pretrained embeddings can boost performance on the relatively small sampled data sets we utilized to create models [9].

2.3 Problem Approach

The predictors we chose are proxies for high-level information and predictions that recruiters or hiring managers find salient in candidate consideration. These include, whether the candidate has enough work experience, how likely they are to stay in the current job and whether they are looking to switch fields.

We focused on benchmarking accuracies from these three types of predictions:

1. Has 10+ years of work experience
2. Stayed at current job next year
3. Switched careers from their last job

Our methodology was to start with an exploratory analysis, ensuring that for unbalanced predictions, we choose the metrics that would accurately gauge performance. In addition, the high salience terms in baseline logistic regression models helped us understand what features are important to each predictor in the dataset.

We sampled our dataset for training sets of roughly 30,000 to 40,000, validating on 10,000 resumes with random resampling. The development set was a separate batch of 50,000 resumes. Originally, we aimed for an order of magnitude larger for training sets, but ran into time and resource limitations when training 50 variants of LSTM networks for each predictor variable.

As baselines, we ran both TFIDF Multinomial naive Bayes and TFIDF logistic regression, which are common and fast baseline models for text classification [10].

Variants of bidirectional LSTM RNN was chosen as the main model of comparison against baselines. Due HTML's grammar of enclosing tags to represent a section, we hypothesized that bidirectional will allow us to capture backward and future context for a HTML sections. LSTM was selected because we wanted to capture long range dependencies, potentially extrapolating conclusions from different sections of the resume.

Because LSTM RNNs can suffer from over training on a smaller dataset, we re-ran the same models across multiple randomly sampled training sets for consistency of results. We also created multiple model variations by tuning input parameters, namely the number of epochs and sequence length.

2.4 Parsing Tools and Data Preparation

Our data set was parsed by BeautifulSoup[3] which helped us standardize and clean HTML as well as extract raw text.

For each resume, we verified that the expected sections exist: education, work experience and has had at least 2 jobs. We also validate that the expected values are complete: years of employment and full job titles for past jobs. Resumes that were incomplete were excluded.

For each of the predictions, we systematically extracted the predictor variable and stripped the resume of that information when applicable. Some of these predictors could have a more nuanced interpretation of actual True or False, but we defaulted to simplified proxies as described below:

1. Has 10+ years of work experience: Resumes were not altered. Boolean predictor was whether the difference between earliest date of their employment history compared to 2017 was greater than or equal to 10.
2. Stayed at current job at year +1: End dates from the current or last position were removed and potential embedded references to current position were also removed. Boolean predictor is True if the end date was 'to present'.
3. Switched careers from their last job: Current or last job title were extracted and removed. Boolean predictor is True if extracted position is similar by counting the number of matching tokens compared to their previous position.

For each of the three predictor variables, we created two training sets, one with validated HTML and one which stripped the HTML but kept the spacing and line breaks.

2.5 Word2Vec Embeddings

[1] https://resumes.indeed.com
[2] https://github.com/eyaler/word2vec-slim
[3] https://pypi.org/project/beautifulsoup4/

Pretrained embeddings such as GloVe, FastText and Word2Vec encode latent relationships between words which can be utilized as part of machine learning model. Training a custom version of Word2Vec on the more resume-specific vocabulary could encode stronger relationships specific to this corpus, but this is a tangential exploration to our project for future exploration.

Because our sampled training dataset was relatively small, we incorporated the pretrained Word2Vec slim embeddings (300k) to our LSTM models to boost training performance. The same embeddings were used across all LSTM models. Embeddings were not updated during training.

2.5 Data Limitations

Our project could have been improved by two major factors:
1. leveraging a substantially larger training set and running in parallel over cloud hosted dedicated RAM/GPU resources
2. leveraging domain specific vocabularies, pre-populating relationships such as from ONET or our own Word2Vec model
3. better and more types of predictors and classifiers, e.g. classification of industry, and running similarity metrics for similar job titles rather than matching tokens

2.6 Metrics

From our exploratory analysis, we have minimal class skew for all our predictors and there's no classifiable substantial difference between false positives or false negative error types. We while tracked f1, loss and absolute accuracy, we opted for absolute accuracy across the test set as the main criteria for model performance. We did investigate a bit into how f1-accuracy differs from HTML and plain text training sets.

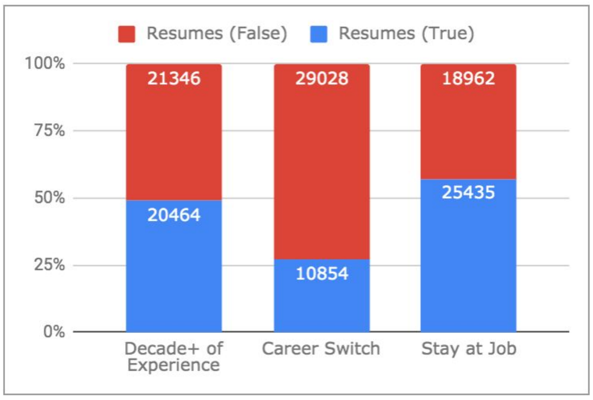


Figure 1: Class Distribution for Predictors

3 Model

3.1 Baseline Models

Given the training datasets we created, we built two baseline models using TFIDF Multinomial Naive Bayes and TFIDF logistic regression, which are common baseline models for text classification [10]. Naive Bayes was able to statistically identify high salience terms while logistic regression serves as a baseline and sanity test, as it is essentially a single neuron neural network with linear decision boundaries. TFIDF is chosen over absolute count vectorization as it more effectively categorizes relevant terms in a document [11].

We created 4 baselines for each predictor: logistic regression and naive Bayes against plain text and HTML decorated resumes.

3.2 Baseline Feature Engineering

In addition to our baseline models, we also attempted rudimentary feature engineering on our dataset to incorporate into our baseline models. Features include: length of the document, average length of tokens, and number of unique tokens. These resume features are shown to have correlation with industry type and length of experience. Although these features are mainly applicable to plain text, we added these features to both plain text and HTML resumes for results completeness.

Logistic regression models that included engineered features did not perform substantially better than vanilla baseline regression models.

3.3 Bidirectional LSTM Setup

The training objective of our model is cross-entropy loss using ADAM optimizer. A recurrent dropout layer was applied and the rate set to 0.1. We used pretrained Word2Vec slim to initialize token embeddings. Words not in the vocabulary were initialized by random normals with mean and standard deviation derived from the loaded embeddings. Batch size was set to 32 examples. The model is implemented using the deep learning framework Keras[4].

3.4 Tuning Bidirectional LSTM

Tuning of our main model was limited to two parameters, number of epochs and sequence length. Due to the smaller training set size and to avoid overfitting, we varied the number of epochs per predictor to retain the best fit model for the development set. We also tuned sequence length because resume text sequences

[4] https://keras.io/

tend to have differing lengths of relevant context. Although 200 to 300 characters is standard for sentences in a news article, we utilized 400 and 800 as an attempt to capture more state in any particular resume section and to address the padded length of HTML context around text.

4 Results

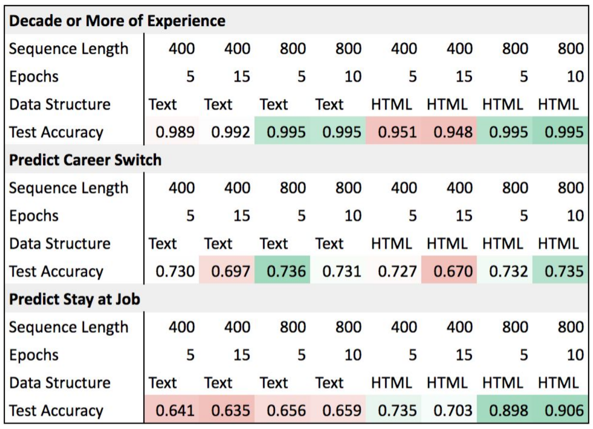
We focused on the absolute accuracy of the test dataset as the primary metric due to a fairly balanced dataset and differentiating between precision and recall is not important. However, we investigated f1-accuracy as a sub-point below as it differs between HTML and plain text LSTM models.

4.1 Analysis of Tuning

The effects of overfitting a small dataset is clearly represented in most of the predictor variables particularly when utilizing a shorter sequence length. For both plain text and HTML, 5 out of 6 test accuracies decreased for a sequence length of 400 from 5 epochs to 15 epochs. This is not true for a longer sequence length of 800, which highlights the importance of tuning dependent input parameters in parallel specific to the dataset.

Using a longer sequence length of 800 also increased the performance of all of the LSTM models for each predictor. This indicates that while part of speech and sentence specific NLP models do well with a shorter sequence length, a longer sequence length is better for long range dependencies in a document.

Although the same parameter values were ran for both plain text and HTML models, it's also clear that plain text and HTML benefit from different values of parameters. For predicting a decade of experience, sequence length strongly impacted HTML's accuracy whereas it did not for plain text. For predicting career switch, additional epochs decreased the accuracy for plain text, but increased the performance of HTML for a sequence length of 800.



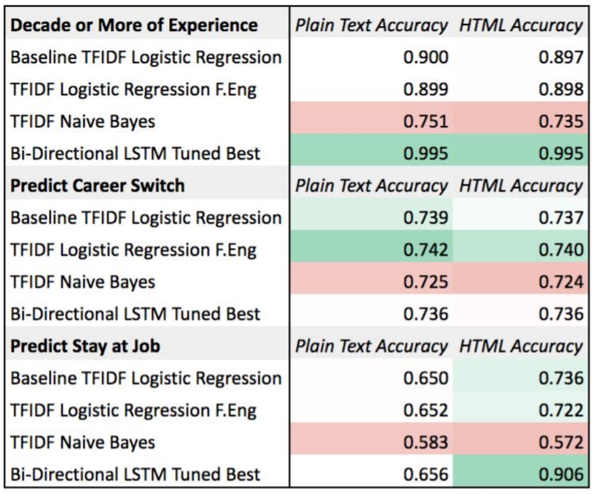
Decade or More of Experience								
Sequence Length	400	400	800	800	400	400	800	800
Epochs	5	15	5	10	5	15	5	10
Data Structure	Text	Text	Text	Text	HTML	HTML	HTML	HTML
Test Accuracy	0.989	0.992	0.995	0.995	0.951	0.948	0.995	0.995
Predict Career Switch								
Sequence Length	400	400	800	800	400	400	800	800
Epochs	5	15	5	10	5	15	5	10
Data Structure	Text	Text	Text	Text	HTML	HTML	HTML	HTML
Test Accuracy	0.730	0.697	0.736	0.731	0.727	0.670	0.732	0.735
Predict Stay at Job								
Sequence Length	400	400	800	800	400	400	800	800
Epochs	5	15	5	10	5	15	5	10
Data Structure	Text	Text	Text	Text	HTML	HTML	HTML	HTML
Test Accuracy	0.641	0.635	0.656	0.659	0.735	0.703	0.898	0.906

Figure 2: LSTM Accuracies Tuning

4.2 Analysis against Baselines

In general, bidirectional LSTM did well compared to the baseline models, with an average increase of 7% accuracy over TFIDF logistic regression across three predictor variables. LSTM was the worst at predicting career switch as it underperformed baselines by a smidgen, accuracy decreased by 0.3% for both plain text and HTML. LSTM saw the best gains in predicting stay at job, accuracy increasing 23% over the baseline.

Comparing results between plain text and HTML models, naive Bayes performed worst in general and worse for HTML models. However, other models generally had comparable performance between the two types of data input (accuracies differed less than 0.5%) the only exception of predicting stay at job. Particularly notable is that HTML data worked the same or better for LSTM models across all three predictors.



Decade or More of Experience	*Plain Text Accuracy*	*HTML Accuracy*
Baseline TFIDF Logistic Regression	0.900	0.897
TFIDF Logistic Regression F.Eng	0.899	0.898
TFIDF Naive Bayes	0.751	0.735
Bi-Directional LSTM Tuned Best	0.995	0.995
Predict Career Switch	*Plain Text Accuracy*	*HTML Accuracy*
Baseline TFIDF Logistic Regression	0.739	0.737
TFIDF Logistic Regression F.Eng	0.742	0.740
TFIDF Naive Bayes	0.725	0.724
Bi-Directional LSTM Tuned Best	0.736	0.736
Predict Stay at Job	*Plain Text Accuracy*	*HTML Accuracy*
Baseline TFIDF Logistic Regression	0.650	0.736
TFIDF Logistic Regression F.Eng	0.652	0.722
TFIDF Naive Bayes	0.583	0.572
Bi-Directional LSTM Tuned Best	0.656	0.906

Figure 3: Model Baselines against Test

4.3 F1 Accuracy on LSTM Models

Although absolute accuracy was similar for predicting decade of experience on LSTM models, the same is not true for f1-accuracy. Accounting for both precision and recall, the LSTM HTML models scored on average 4% better on f1-accuracy than plain text. Comparing confusion matrix for the best LSTM models on predicting experience, we see that not only is HTML a fraction more accurate, but also it's balances false positives and false negatives better than plain text.

Decade or More of Experience	*F1 Plain Text*	*F1 HTML*
LSTM SeqLen:400 Epochs:5	0.796	0.793
LSTM SeqLen:800 Epochs:5	0.799	0.815
LSTM SeqLen:800 Epochs:10	0.798	0.877
LSTM SeqLen:400 Epochs:15	0.808	0.846

Figure 4a: F1-Accuracy

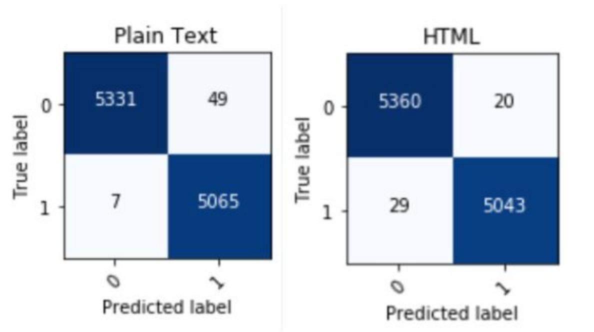


Figure 4b: Confusion Matrix for Plain, HTML - Years of Exp

4.4 Analysis of Predictors and Errors

Among the three predictors, predicting career switchers had highest error rate for LSTMs. Partly due to the inherent difficulty of the task, error analysis showed that even humans would have a hard time correctly identifying the cases where LSTM had failed. The most salient terms identified by the Logistic regression baseline consisted of specific job titles, e.g. 'nurse', 'truck driver', 'teacher' and career attributes e.g. 'senior' and 'certified'. We conjecture that logistic regression has very similar performance to LSTM partly due to linear relationships for specific job function retention.

Classifying whether someone had a decade more of work experience had a fairly high accuracy baseline of 90%, where salient terms identified from logistic regression were individual years previous to this decade. However, where LSTM improved significantly was the ability to identify work experience dates separately from dates in education, certifications or skill, e.g. 'SQL Server 2005' and reach a near perfect accuracy rate. The best model was LSTM with HTML text at sequence length 800 running 10 epochs, which outperformed plain text by a fraction of a percent.

HTML injected LSTMs had the most success predicting whether someone stayed at the same job following year, a 23% increase from baseline and 38% increase from plain text models. Confusion matrices show that HTML performed better in all four quadrants and excelled in true negatives, where it correctly identified 50% more folks who left their current job. However, the confidence in these predictions were relatively low. We hypothesize from error analysis that the HTML LSTM was able to identify the summary and current work experience sections which would contain the more relevant information on whether someone stayed at their current job.

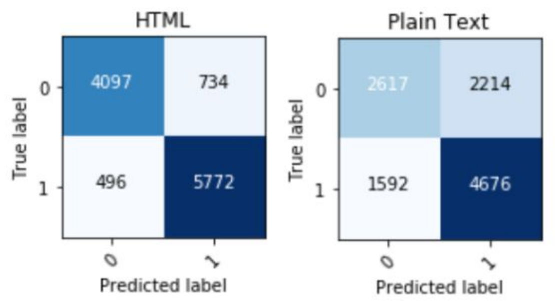


Figure 5: Confusion Matrix for HTML, Plain - Stay at Job

5 Conclusion

We demonstrated the viability of using structured HTML in bidirectional LSTM RNN models a resume corpus for prediction and classification. Model features included pre-trained Word2Vec embeddings and features generated by the RNN. While LSTM was either on par or significantly better than baseline logistic regression models, using HTML in the neural models also demonstrated performance on par or better than plain text.

Despite the positive results, utilizing larger training sets, creating a custom trained set of embeddings from the resume corpus, and further tuning of LSTM input parameters would likely improve performance for each of the predictor variables we tested.

References

[1] H. Wang, X. Zhang, A Neural Question Answering Model Based on Semi-Structured Tab
http://aclweb.org/anthology/C18-1165
[2] J. Krishnamurthy, P. Dasigi, and M. Gardner, Neural Semantic Parsing with Type Constraints for Semi-Structured Tables
https://www.aclweb.org/anthology/D17-1160
[3] G. Qin, J.G. Yao, Learning Latent Semantic Annotations for Grounding Natural Language to Structured Data
http://aclweb.org/anthology/D18-1411
[4] T. Gogar, O. Hubacek, J. Sedivy, Deep Neural Networks for Web Page Information Extraction

https://hal.inria.fr/hal-01557648/document
[5] A. García-Plaza, V. Fresno, Using Fuzzy Logic to Leverage HTML Markup for Web Page Representation
https://ieeexplore.ieee.org/abstract/document/7505655
[6] E. Angelino, Extracting Structure from human-readable Semistructured Text
https://people.eecs.berkeley.edu/~elaine/pubs/angelino-structure.pdf
[7] A. Hotho, S. Staab, G. Stumme, Text Clustering Based on Background Knowledge
https://pdfs.semanticscholar.org/56d0/7518b3a83229656a993adf0dd64ba26da3a6.pdf
[8] Wikipedia: Semi-structured data
https://en.m.wikipedia.org/wiki/Semi-structured_data
[9] Y. Qi, D. Sachan, When and Why are Pre-trained Word Embeddings Useful for Neural Machine Translation?
https://www.aclweb.org/anthology/N18-2084
[10] S. Wang, C. Manning, Baselines and bigrams: simple, good sentiment and topic classification
https://dl.acm.org/citation.cfm?id=2390688
[11] J. Ramos, Using TF-IDF to Determine Word Relevance in Document Queries
https://www.semanticscholar.org/paper/Using-TF-IDF-to-Determine-Word-Relevance-in-Queries-Ramos/b3bf6373ff41a115197cb5b30e57830c16130c2c