CS230

Predicting Loan Grades using Deep Neural Networks

Kaushal Alate
Department of Computer Science
Stanford University
kalate@stanford.edu

Bilguunzaya Battogtokh
Department of Computer Science
Stanford University
bbatt99@stanford.edu

Liang Ping Koh
Department of Computer Science
Stanford University
lpkoh@stanford.edu

Abstract

Assigning grades to loans is an important task for lending institutions to manage risk. However, this task can be time-consuming and expensive to do manually. This project aims to automate this process by training and evaluating a neural network model on a dataset of labelled loans from the online peer-to-peer lending platform LendingClub. Our approach builds on existing work by augmenting the input with text embeddings of text features such as loan description. We use both pre-trained Word2Vec word embeddings and custom-trained Word2Vec embeddings. Text embeddings do not seem to improve accuracy, a finding that will inform future work on this task.

1 Introduction

Lenders assign grades to loans to help them manage their overall risk, set interest rates etc. Assigning grades can be expensive, difficult and time-consuming as the human grader has to consider many variables such as the borrower's background and credit history. An automatic system to predict loan grades would improve a financial institution's efficiency and/or be a valuable verification tool for human-assigned grades. The input to our models consists of the borrower's socioeconomic information such as income, loan amount requested, home ownership, debt-to-income ratio etc. as well as three borrower-provided text features: 1) loan description, 2) loan title and 3) employment title of borrower. The output of our model is the loan grade (one of A, B, ..., G, where A is the best grade).

Kaushal Alate worked on a project predicting loan grades in CS221 in Fall 2018. This project takes inspiration from that CS221 paper (which involved logistic regression, k-means clustering and a simple neural network) but the neural network here has more hidden units, and uses word embeddings to include the three text features mentioned above. The inclusion of more numerical and categorical features results in a much higher accuracy of 90.2% for loan grade prediction.

2 Related work

2.1 Predicting Loan Grades with a Neural Network: A Machine Learning Pipeline on AWS

This model, also trained on LendingClub data, is our baseline, though we have made many modifications as described under "Dataset and Features" and "Models". It is a two hidden layer neural net but without any of the text embeddings we use and without much hyperparameter tuning. It achieves 87.6% accuracy and is thus quite successful. Our work builds on this model by integrating text embeddings for the loan "description", "title" and (borrower's) "employment_title" text fields.

2.2 A Neural Network Approach for Credit Risk Evaluation.

Using data from a bank in Italy, the authors used a feed-forward neural network with two hidden layers to predict whether a small business loan will default. For their features, they focused on data specific to the borrower, such as the borrower's credit lines, total assets, etc. It was found in the paper that macroeconomic variables, such as business environment, had a small impact on small to middle size loans. Since we were working with relatively small loans made to individuals in our dataset, we followed this paper's advice and did not focus on adding in new macroeconomic variables. This paper's network architecture, with two hidden layers, corroborates what we used. It was able to achieve a test accuracy of 91.4%.

2.3 Bankruptcy Prediction for Credit Risk Using Neural Networks: A Survey and New Results.

The authors of the paper improved their accuracy in predicting corporate bankruptcies from 81.46% to 85.5% (out of sample) for their neural net system by engineering novel new features, such as stock price volatility and rate of change of cash flow per share. We could not have engineered these specific features as the loans in our dataset are for individuals, but future work may involve creating new features in a similar manner.

2.4 A data-driven approach to predict default risk of loan for online Peer-to-Peer (P2P) lending.

This paper uses two decision trees, two neural networks and one SVM to predict loan default. It concludes that the term of loan, borrower's annual income, amount of loan, debt-to-income ratio, credit grade and revolving line utilization are important when predicting loan default. We infer that these are likely also significant when predicting loan grades and have included these in our model.

2.5 The use of profit scoring as an alternative to credit scoring systems in peer-to-peer (P2P) lending.

This paper performs both a survival analysis and a logistic regression analysis for each of the different features. The survival analysis generates for different features a survival curve, which give the probability of default at certain points in time. This could be a useful reference for future projects that seek to predict loan risks over time. The report used seven logistic regressions using individual features to better understand the predictive ability of each of the individual features, an approach we could consider in future work for our dataset as well.

3 Dataset and Features

The dataset had many missing values, so our first task was to remove certain features (columns) and loans (rows) so that we could have a dataset of complete rows. We decided to remove feature columns which had more than 50% missing values and we then removed rows which had any incomplete values. The choice of 50% worked well because this left us with a dataset with a good size of 63,207 loans. All categorical features, such as home ownership status, are then converted to one-hot encodings. All numerical features were normalized by subtracting each feature's mean from the values and dividing the difference by the feature's standard deviation. The training/validation/testing split is 60/20/20% which gives 12641 loans for validation and testing each, which was thought to be sufficient.

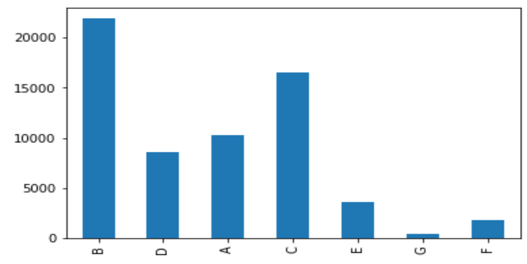


Figure 1: Histogram showing the distribution of loan grades among the loans

4 Methods

4.1 Model without text embeddings

This model is similar to the baseline (Andersen) model but we performed some tuning (or verification) of the hyperparameters. There are two hidden layers, both with dropout with a rate of 20% and ReLU activation functions. The Andersen paper used regularization (the norm was capped at 3), which we also did not change because overfitting (training error > valid error) was observed without regularization. The Adam optimizer learning rate and the number of units in the first layer were kept at 0.001 and 100 respectively, but the number of hidden units in the second layer was reduced from 60 to 50 (a random search strategy was used). A minibatch size of 1024 was used to make training fast and reducing the minibatch size did not seem to increase the accuracy.

4.2 Model with text embeddings (computed using pre-trained Word2Vec word embeddings)

This model added text embeddings for the description, title and employment title text features. We explain how these were calculated and integrated into the neural net. We tokenized each string (description, title or employment title) by dividing on white spaces, took the lowercase of each word and removed non-alphanumeric characters. We then found the mean of the pretrained Word2Vec OntoNotes word embeddings for each word that is present in the string and in the the OntoNotes embeddings dictionary, and used PCA to reduce this 300-dimensional mean vector to 50 dimensions for faster training.

Three such 50-dimensional embeddings were calculated for each loan (for description, title and employment title). These were concatenated with the input layer, increasing the dimensionality of the input layer by 150. The model with two hidden layers (with the same number of hidden units, dropout and activation functions as in model 1) was then trained. Random search again led to hyperparameter values of 0.001 for the Adam optimizer learning rate, and 1024 for the minibatch size was used as reducing it further did not seem to improve accuracy.

4.3 Model with text embeddings using custom-trained Word2Vec word embeddings

This model added custom-trained embeddings for the description, title and employment title text features to model 1. First, we explain how the custom-trained Word2Vec embeddings were computed. Briefly put, Word2Vec computes vector embeddings of words using gradient descent by seeking to maximize the dot product of similar words. Similar words are words that occur within each other's "context" in a sentence, where the context of a word is the words within a fixed distance of the word (we used the gensim package's default distance of 5 words).

To train our word embeddings, we had to start by extracting sentences from our textual data. We used gensim's sentence extraction tool (whose main method seems to be splitting on periods). We then tokenized the words of each sentence, converted to lowercase and removed non-alphanumeric characters. Using these processed "sentences" from the loan description, title and employment title fields, we used gensim to train our word embeddings, which we chose to make 50-dimensional for fast training.

Like in model 2, we used these custom-trained word embeddings to compute a text embedding for each loan's description, title and employment title. We again used the tokenized lowercase and non-alphanumerics-removed form of each string. We then found the mean of the custom-trained word embeddings for each word that is present in the string and in the dictionary mapping words (present in the training data) to the custom-trained embeddings.

Again, three such 50-dimensional embeddings were calculated for each loan (for description, title and employment title) and were concatenated with the input layer. The other hyperparameters were kept the same as in model 1 as trying other values using random search did not lead to noticeable accuracy improvements, except for minibatch size which was decreased to 256 as this led to higher accuracy.

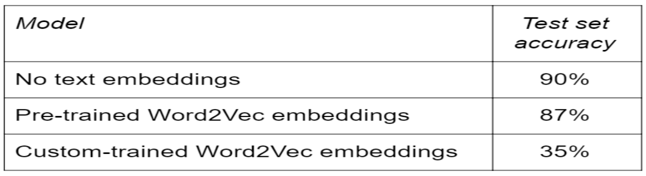
The loss function for all three models was standard categorical cross-entropy loss, given by:

$$CE = -\sum_{i}^{C} t_i log(s_i)$$

Here t_i is the true value and s_i is the score for each class i.

5 Experiments/Results/Discussion

Since there are 7 classes (grades can be A, B, ..., F or G), accuracy, or the percentage of loans classified correctly, was used as the evaluation metric and the results for the three models are summarized below:



Model	*Test set accuracy*
No text embeddings	90%
Pre-trained Word2Vec embeddings	87%
Custom-trained Word2Vec embeddings	35%

Figure 2: Table of accuracy for each model.

The graphs of loss falling over epochs are shown below for each of the three models. Each model was trained for the number of epochs needed for loss to plateau (50, 100 and 10 epochs for models 1, 2 and 3 respectively).

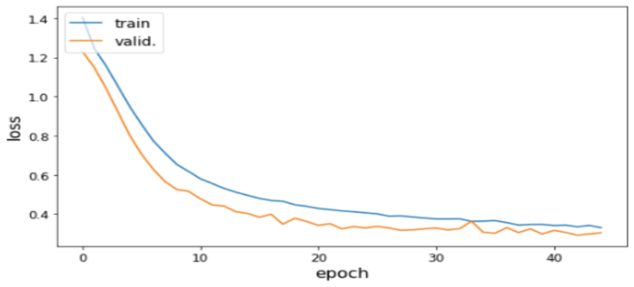


Figure 3: Model 1: No text embeddings

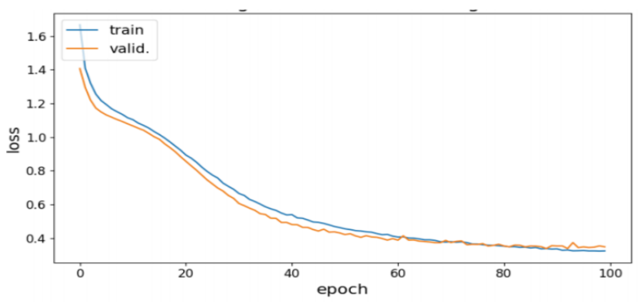


Figure 4: Model 2: Pre-trained Word2Vec embeddings

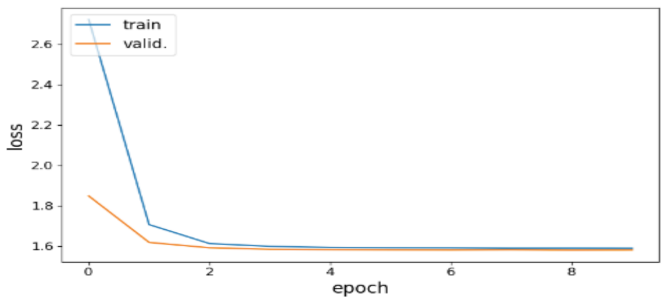


Figure 5: Model 3: Custom-trained Word2Vec embeddings

The neural network with no embeddings has the highest test accuracy of 90.2%. Integrating text embeddings based on pre-trained word embeddings led to a test accuracy of 87%, whereas using custom-trained word embeddings led to a test accuracy of 35%. Using custom-trained Word2Vec embeddings may have resulted in a severe drop in accuracy because of the relatively small quantity of data available for training such embeddings – each of the 63,207 loans has only at most a few sentences of description and a few words for title and employment title.

An example of an incorrectly classified loan is a D-grade loan that was classified as a B by the no-embeddings model. It is difficult to analyze exactly why this misclassification occurred but it may be due to the fact that the borrower's annual income is quite high ($120,000) and the borrower owns a home despite their debt-to-income ratio being quite high (16.57).

Another example is an A-grade loan that was classified as a B by the no-embeddings model. Again, it is difficult to say why but the debt-to-income ratio here is rather high (8.38), although the loan amount is relatively small ($12,000) compared to the borrower's high annual income ($140,000).

6 Conclusion/Future Work

This project considered the effect of adding text embeddings to the existing model for predicting loan grades. It seems that using text embeddings does not increase the accuracy of the existing model, which is a valuable finding as it will inform future research on the topic. The relatively small quantity of data available for training custom embeddings may be responsible for their low performance. We were able to replicate the high accuracy of the model by James Andersen.

Future work on this task could involve trying more sophisticated computation methods for text embeddings than finding the mean of the word embeddings – for instance, an RNN could be used with the different words' embeddings as inputs. It would probably be better to use pre-trained word embeddings as there does not seem to be sufficient data for custom-training embeddings.

We might also explore different network topologies. In Angelini et al., the neural net had input neurons being grouped into triplets and each triplet being connected to only one neuron in the first hidden layer, which led to better test set error (4.3% as opposed to 8.6%), suggesting that instead of fully connected layers, more restricted connections could perform better.We could also expand the task to not just predict loan grade at one point in time, but to model the evolution of a loan's risk over a time period. For this, we would need to find data on loans over a period of time and use RNNs.

7 Contributions

All team members made a significant contribution to this paper. Kaushal Alate worked on training custom word embeddings, integrating these into the neural network and writing the final report. Bilguunzaya Battogtokh was involved with integrating pre-trained word embeddings into the neural network, creating visualizations and preparing the poster. Liang Ping Koh was involved with integrating pre-trained word embeddings into the neural network, dimensionality reduction techniques and writing the final report.

8 Github Repository

Link: https://github.com/kalate/CS230-loan-grades

References

[1] Andersen, James. (8 Sept. 2017) Predicting Loan Grades with a Neural Network: A Machine Learning Pipeline on AWS. *Medium*.

[2] Angelini, Eliana, et al. (Nov. 2008) "A Neural Network Approach for Credit Risk Evaluation." *The Quarterly Review of Economics and Finance*.

[3] Atiya, Amir (July 2001) "Bankruptcy Prediction for Credit Risk Using Neural Networks: A Survey and New Results." *IEEE Transactions on Neural Networks*.

[4] Jin, Yu, and Yudan Zhu (2015) "A data-driven approach to predict default risk of loan for online Peer-to-Peer (P2P) lending." *2015 Fifth International Conference on Communication Systems and Network Technologies*.

[5] Serrano-Cinca, Carlos, and Begoña Gutiérrez-Nieto (2016) "The use of profit scoring as an alternative to credit scoring systems in peer-to-peer (P2P) lending." *Decision Support Systems 89*.

9 Libraries

Code libraries used: Keras, Pandas, Numpy, Sklearn, Matplotlib, Seaborn, Gensim, Datetime, csv