Which attributes matter the most for loan origination? A neural attention approach

Antonios Alexos
Department of Electrical and Computer Engineering
University of Thessaly
Greece, Volos
aalexos@uth.gr

Sotirios Chatzis
Cyprus University of Technology
Department of Electrical Engineering, Computer Engineering and Informatics
Cyprus, Limassol
sotirios.chatzis@cut.ac.cy

Abstract

In this paper we address the understanding of the problem, of why a deep learning model decides that an individual is eligible for a loan or not. Here we propose a novel approach for inferring, which attributes matter the most, for making a decision in each specific individual case. Specifically we leverage concepts from neural attention to devise a novel feature wise attention mechanism. As we show, using real world datasets, our approach offers unique insights into the importance of various features, by producing a decision explanation for each specific loan case. At the same time, we observe that our novel mechanism, generates decisions which are much closer to the decisions generated by human experts, compared to the existent competitors.

1 Introduction

A very important question that we must consider nowadays, is who will take the decisions regarding loans? Humans that work for the bank, or algorithms? Every decision must be treated with fairness; to ensure fairness, explainability is the key. Thus we are going to talk about how to achieve this fairness in loan decisions.

In order to move to this direction we use the concept of neural attention. We want to explain each loan decision in the best possible way, with unique insight for every individual application, so we take advantage of the Neural Attention in an Encoder-Decoder model. Currently neural attention is used in Encoder-Decoder models for sequence to sequence inputs, in order to put emphasis in different time frames. In contrast to this, we use neural attention in order to infer the importance of each decision in each specific case of loan application. Then, we utilise this information, so that we can better exploit the existent information, to better inform the decision making of the model. So in this paper, we are trying to show another side of these models, the one that focuses on the attention mechanism, and thus extracting the feature importance of the data Ning Gui (2019).

We evaluate this proposed method with data found from the Home Mortgage Disclosure Act(HMDA). We compare this method, with existent methods like Decision Trees, Lasso, etc and we prove its uniqueness and great performance. This method achieved a **89.6%** prediction score.

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

The rest of the paper is organised as follows. Section 2 provides a short introduction to neural attention for sequence to sequence models. Section 3 shows our approach, while section 4 shows the Experimental Setup for our experiments. Moreover section 5 is about the prediction results, and section 6 is about feature importance and more results. Finally, the conclusions and future work are drawn in the last section of the paper.

2 An introduction to Neural Attention for Sequence to Sequence modeling

The Encoder-Decoder architecture that we used in this paper is adopted from a tutorials Luong et al. (2017). In this model, we consider a deep RNN layer, and we use as a recurrent unit, a gated recurrent unit(GRU) Chung et al. (2014). The GRU is very similar to a long short-term memory (LSTM) Hochreiter and Schmidhuber (1997), but it has a forget gate and fewer parameters than LSTM, as it lacks an output gate.

So this model consists of two recurrent neural networks: the encoder RNN simply consumes the input data without making any predictions; and the decoder RNN, which processes the target making the prediction for the loan application. For more information about how the encoder and decoder were built, you can refer to Luong et al. (2015).

The attention mechanism that we use, was first introduced by Bahdanau et al. (2014). Here, every decoder output is regarded based on some previous outputs and some $\mathbf{x}$, where $\mathbf{x}$ consists of the current hidden state and the attention "context". So the decoder defines a probability over the result $\mathbf{y}$ by decomposing the joint probability into the ordered conditionals:

$$p(\mathbf{y}) = \prod_{t=1}^{T} p\left(y_t \middle| \left\{y_1, \cdots, y_{t-1}\right\}, c\right) \tag{1}$$

where $y_{t'}$ is the next prediction and$\{y_1, \cdots, y_{t'-1}\}$ are all the previous predictions that the model made; c is the context vector, and $\mathbf{y} = \left(y_1, \cdots, y_{T_y}\right)$. With an RNN, each conditional probability is modeled as:

$$p\left(y_i | y_1, \ldots, y_{i-1}, \mathbf{x}\right) = g\left(y_{i-1}, s_i, c_i\right) \tag{2}$$

where g is a nonlinear function that outputs the probability of y_t , and s_i is the current hidden state calculated by an RNN f with the last hidden state s_{i-1}, computed by:

$$s_i = f\left(s_{i-1}, y_{i-1}, c_i\right) \tag{3}$$

The context vector c_i depends on a sequence of annotations $(h_1, \cdots, h_{T_x})$ to which an encoder maps the input sentence. Each annotation h_i contains information about the whole input sequence with a strong focus on the parts surrounding the i -th part of the input sequence.

The context vector c_i is a weighted sum of all encoder outputs, where each weight a_{ij} is the amount of "attention" paid to the corresponding encoder output h_j .

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j \tag{4}$$

The weight α_{ij} of each annotation h_j is normalized and computed by

$$\alpha_{ij} = \frac{\exp\left(e_{ij}\right)}{\sum_{k=1}^{T_x} \exp\left(e_{ik}\right)} \tag{5}$$

where

$$e_{ij} = a\left(s_{i-1}, h_j\right) \tag{6}$$

is an alignment model which scores how well the inputs around position j and the output at position i match. The score is based on the RNN hidden state S_{i-1} and the j -th annotation h_j of the input sentence.

3 Our Approach

It is generally known that Encoder-Decoder models take as an input, sequences of vectors, and produce as an output, also sequences of vectors. In our work, we treat every observation(vector) loan data as a sequence of tokens, with each feature being treated as a token by the model.

We make the assumption that the data are not sequential, and do not have any short-term dependencies. This creates a slight violation to the model. By making this assumption we explicitly consider sequential dynamics. The form of dynamics of dependencies is adaptive due to the attention mechanism. So this slight violation that exists, is not detrimental due to the attention mechanism. We believe, that by using the attention mechanism, we compensate for this assumption, of not having sequential dependencies. The attention mechanism allows the model to see long-term dependencies. This relaxes the strict sequential assumption and allows the model to form any kind of dependencies in the vector. And in a dynamic fashion, because different observations lead to different kinds of dependencies.

So each observation has 26 features, which we put into our model. Then we put every observation into the encoder where it passes from 2 layers. The first one is an Embedding layer and the second one is a simple GRU layer. We use a GRU as we assume that there is not a strict ordering of the observed features, but there are dependencies between them. Afterwards the data pass to the Decoder, and firstly pass through an Attention Layer by Bahdanau et al. (2014). The attention weights are to an Embedding Layer, which then passes the data to the GRU. Eventually, the produced content vector is presented to a Decoder which is configured, in the same way as the Encoder(but with a Dense layer before the output), to produce a sequence of length three. Thus, the output is a sequence of 3 probabilities. Therefore our objective function aims to get each target value of the three produced outputs as close to 0 or 1 as possible. We take then the probability of the mean of these three numbers and assign it to either 0 or 1. We did not choose the output to be three for a specific reason. We tried also combinations of the output to be two and four, but it did not make a lot of difference on the result.

During the training of the model we use the Mean Squared Error loss function, which takes the mean of the output and compares it with the target which is either 0 or 1. In this way our model manages to get optimised. Neural Attention helps us extract the attention weights and present them to the client in an user-friendly way, which will explain how the algorithm took the decision for the application of the user. So we extracted the attention weights and plotted them in a heatmap, which can be shows to a bank client in a user-friendly way.

You can see an overview of our model in Figure 1.

4 Experimental Setup

The data that was used in this paper is are from the HMDA. Currently according to their website, they are the most comprehensive source of publicly available information on the U.S. mortgage market. The data set contains 466,566 observations of Washington State home loans. The subset that we used has 10% of the dataset, 46,657 observations. The main reason that we chose this data set for this paper, is because it contains sensitive data; data like applicant race name, applicant ethnicity name, sex name, co-applicant race name, co-applicant ethnicity name. Chiefly for experimenting reasons we didn't test the whole data set, but we took a small portion from it; 10% of the data. After that we wanted to keep only the observations that had to do with the primary market, where only borrowers and lenders are involved. So we dropped the rows that had on column action-taken-name either "Application withdrawn by applicant" or "Loan purchased by the institution". Also we dropped almost half of the columns because they were mostly empty, in order to make the data set less sparse.

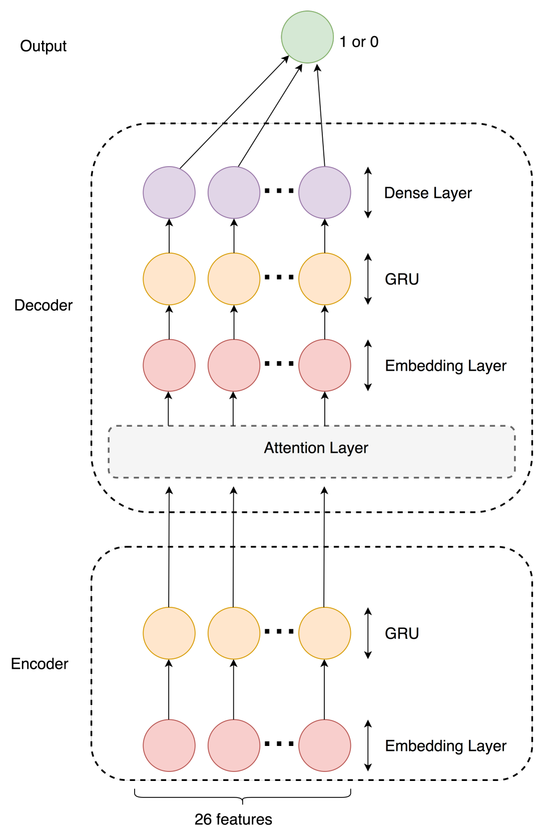


Figure 1: Overview of our model

[1] In the next step we transform all of the data to integers and split the data set to 80-20% train-test. After all the preprocessing, the final data set that we use in this paper, has 26 features.

5 Prediction Results

In this section we are going to present the prediction of the Encoder-Decoder with Attention model, regarding that this is a simple binary classification problem; 1 for loan approval and 0 for denial. In order to prove the credibility and the superiority of the model, we compare its results, with the results of a simple neural network. On figure (2) we can see the summary of the simple model that we have built:

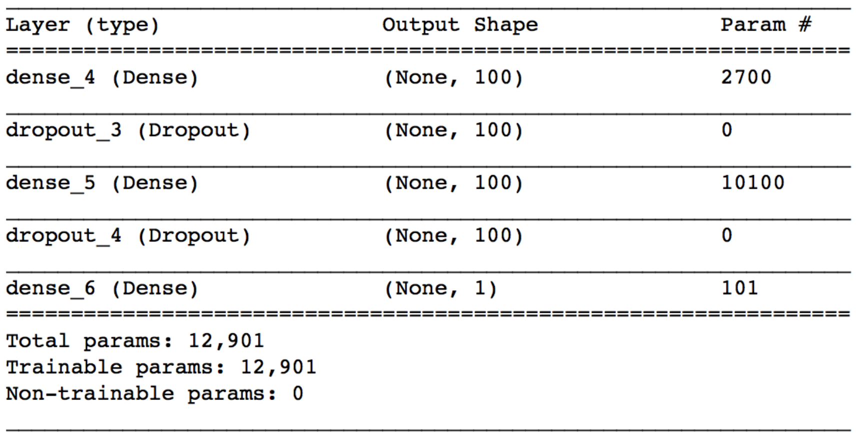
As we can see the neural network that we have built is really simple, and it doesn't include any advanced layers. Let's observe now the results of the prediction:

This produces a prediction score of **73.2%**.

Regarding the Encoder-Decoder with Attention model we get the confusion matrix in figure (4):

The model produces a prediction score of **89.6%**. Comparing these two models we can see that the Encoder-Decoder with Attention model performs better, both in the confusion matrix and the prediction score. This is valid proof that our model works better than a simple neural network.

[1]Some of the columns that we dropped are rate_spread, msamd-name, respondent-id, application-date-indicator, the co-applicant-race-name columns that are above 1, as well as the applicant-race-name columns that are above 1(for example applicant-race-name-2, applicant-race-name-3, applicant-race-name-4, applicant-race-name-5, etc).



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_4 (Dense)              (None, 100)               2700
_________________________________________________________________
dropout_3 (Dropout)          (None, 100)               0
_________________________________________________________________
dense_5 (Dense)              (None, 100)               10100
_________________________________________________________________
dropout_4 (Dropout)          (None, 100)               0
_________________________________________________________________
dense_6 (Dense)              (None, 1)                 101
=================================================================
Total params: 12,901
Trainable params: 12,901
Non-trainable params: 0
_________________________________________________________________
```

Figure 2: Summary of the simple neural network

	Yes	No
Yes	0	1914
No	0	5243

Figure 3: Confusion matrix of the simple neural network

	Yes	No
Yes	1307	559
No	179	5059

Figure 4: Confusion matrix of the model

6 Feature Importance

In this section we will present the feature importance results, which the Encoder-Decoder with Attention model produced, and compare them with the results of classical algorithms for feature importance. Firstly, we are going to present the other algorithms and their results, and afterwards we are going to present the results of the Encoder-Decoder model. We are not going to get into detail for the other algorithms, so we will just present an introduction for every one of them, and refer to their original publications.

6.1 XGBoost

XGBoost stands for eXtreme Gradient Boosting, and it is an optimized distributed gradient boosted library designed to be highly efficient, flexible and portable Chen and Guestrin (2016). XGBoost provides a parallel tree boosting which solves many data science problems in a fast and accurate way. An advantage of XGBoost is that after the construction of the boosted trees, it is very easy to retrieve the feature importance the importance score for each feature. The importance score for each feature, shows us how useful the feature is regarding the construction of the trees inside the model. The importance score is calculated for a tree by the amount that each feature point split improves the measure of performance for the model, regarding the observations, which the node is responsible for. The measure of performance can be of course the Gini index, which selects the split points. This model basically ranks the features of the data set comparing them to each other. This algorithm achieces an a **89%** accuracy score. You can see the feature importance heatmap on Figure(5).

doi:10.20944/preprints202002.0180.v1

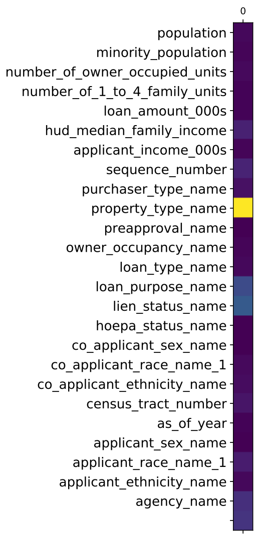


Figure 5: Feature Importance of XGBoost

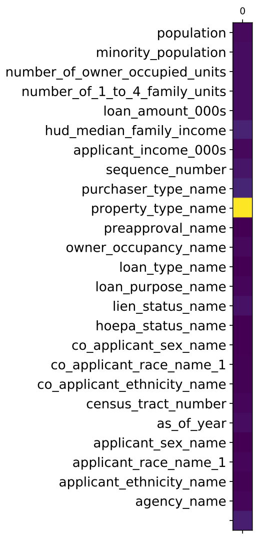


Figure 6: Feature Importance of Decision Tree

6.2 Decision Tree

Decision Tree Quinlan (1986) is a supervised learning method used mostly for classification. The way that this method learns from data, is achieved with a set of if-then-else decision rule statements; thus, the deeper the tree, the more complex are the decision rules of it. Like its name, a Decision Tree is a tree structure, and it breaks down a data set intro smaller subsets, while at the same time a decision tree associated with the smaller subsets is also created, and produces as a result, a tree with decision nodes and leaf nodes. Each of these nodes has two or more branches, while every leaf node is either a classification or a decision. This method achieved an **84.7%** accuracy on the problem. You can see the feature importance heatmap on Figure(6).

6.3 Extra Trees Classifier

Extra Trees Classifier, also known as Extremely Randomized Trees, is an ensemble method, based on Random Forest Geurts et al. (2006). They both build multiple trees and split nodes, but the difference is that Extra Trees pick random decision boundaries at each step of the iteration, while the entire sample data set is used. This method achieved an **88%** accuracy score. You can see the feature importance heatmap on Figure(8).

6.4 LASSO Regression

LASSO is a regression model Tibshirani (1994), which involves a penalty factor that determines how many features are retained; using cross-validation to choose the penalty factor helps assure that the model will generalize well to future data samples. In more detail, LASSO penalizes the l_1 norm of the weights, which induces sparsity in the solution; many weights are forced to zero, or really close to zero.

This performs variable selection (the ’relevant’ variables are allowed to have nonzero weights). The degree of sparsity is controlled by the penality term, and some procedure must be used to select it. On each iteration, the next best regressor is added to the active set. Then, the weights for all regressors in the active set are recomputed. Because of the reweighting step, this approach is less greedy (and has better performance) than the regular matching pursuit/stepwise regression. But, it still employs a greedy search heuristic, thus, making the fitted model more interpretable. You can see the feature importance heatmap on Figure(7).

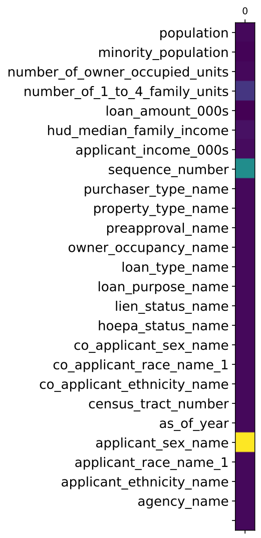


Figure 7: Feature Importance of LASSO

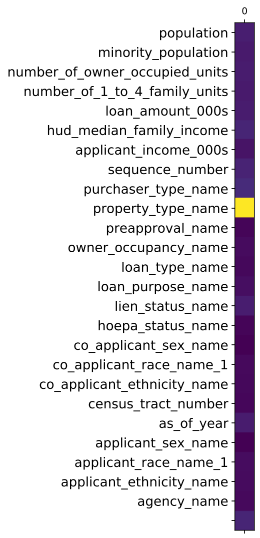


Figure 8: Feature Importance of Extra Trees

6.5 Encoder-Decoder with Attention

Here are the results of our approach, which we have explained in great detail in section 3. Here we can see the uniqueness of the result in comparison with the other models. Our approach produces a unique feature importance for every application. We can see that on the heatmaps of attention weights, of 4 different loan applications, on Figures9, 10, 11, 12.

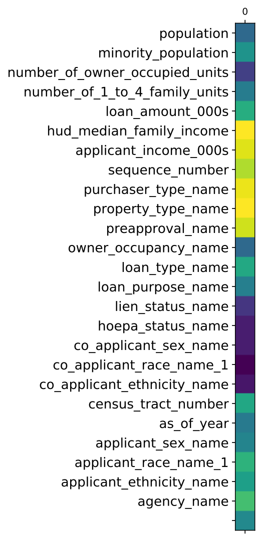


Figure 9: Feature Importance of application 0

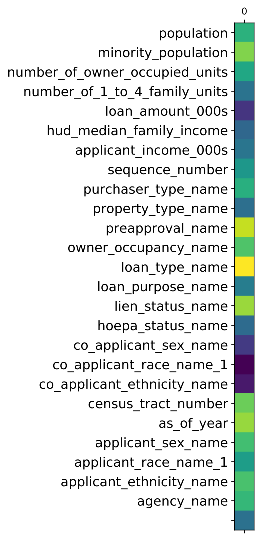


Figure 10: Feature Importance of application 1

7 Conclusions and Discussion

We have introduced a different aspect of the Encoder-Decoder with Attention model, and a side of this model, that is not widely known. This model has been compared with a simple neural network for its predictions, where it shows its superiority. It has also been compared with other algorithms for its feature importance and accuracy, where the Encoder-Decoder with Attention model has shown its uniqueness and superiority. This model has clearly other utilities apart from translating sentences in other languages. It has the ability to show unique feature importance for every observation. In this way we have successfully achieved, to explain bank loan decisions to the applicants, by giving them unique explanations regarding their application. Apart from that, through data analysis we have seen that the algorithm is not discriminating in the rejection cases. Features like race and ethnicity do not play a major role in a rejected case. Regarding our future work, our goal is to implement BERT on our data, and hopefully see some improvements. We can also experiment with changing the order of the features, and observe any changes on the accuracy. Afterwards, we could try to test the model, with the whole dataset in order to produce a better prediction. Another idea would be to implement the model to other data than the ones presented here.

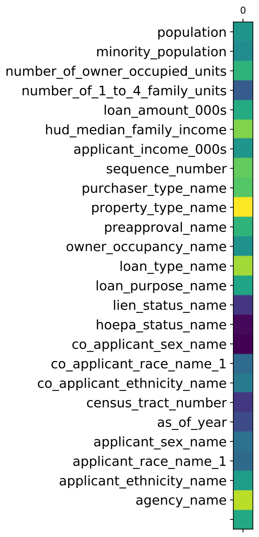


Figure 11: Feature Importance of application 2

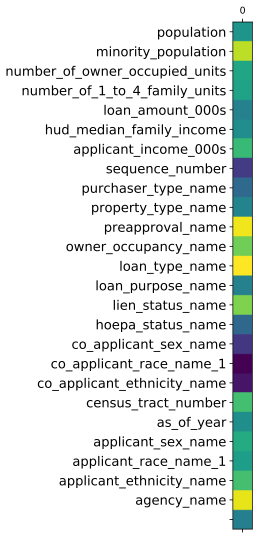


Figure 12: Feature Importance of application 3

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate.

Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, New York, NY, USA, KDD '16, pages 785–794.

Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *NIPS 2014 Workshop on Deep Learning, December 2014*.

Pierre Geurts, Damien Ernst, and Louis Wehenkel. 2006. Extremely randomized trees. *Mach. Learn.* 63(1):3–42.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.* 9(8):1735–1780.

Minh-Thang Luong, Eugene Brevdo, and Rui Zhao. 2017. Neural machine translation (seq2seq) tutorial. *https://github.com/tensorflow/nmt* .

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *EMNLP*.

Danni Ge Ziyin Hu Ning Gui. 2019. Afs: An attention-based mechanism for supervised feature selection. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence*.

J. R. Quinlan. 1986. Induction of decision trees. *Mach. Learn.* 1(1):81–106.

Robert Tibshirani. 1994. Regression shrinkage and selection via the lasso. *JOURNAL OF THE ROYAL STATISTICAL SOCIETY, SERIES B* 58:267–288.