Predicting Customer Churn in the Telecommunication Industry by Analyzing Phone Call Transcripts with Convolutional Neural Networks

Conference Paper · March 2019

DOI: 10.1145/3319921.3319937

2 authors, including:

Junmei Zhong
Marchex
23 PUBLICATIONS **185** CITATIONS

SEE PROFILE

Predicting Customer Churn in the Telecommunication Industry by Analyzing Phone Call Transcripts with Convolutional Neural Networks

Junmei Zhong
Marchex Inc
Suite 2000, 520 Pike Street,
Seattle, WA, USA 98101
jzhong@marchex.com

William Li
Marchex Inc
Suite 2000, 520 Pike Street,
Seattle, WA USA 98101
wli@marchex.com

ABSTRACT
For telecommunication service providers, a principle method for reducing costs and generating revenue is to focus on retaining existing customers rather than acquiring new customers. To support this strategy, it is important to understand customer concerns as early as possible to prevent churn: The customer action of canceling a subscription and moving to a new provider. In this paper, we use actual customer phone call data and develop the convolutional neural network (CNN)-based predictive model to detect churn signals from transcript data of phone calls. Experimental results show that when sufficient training data is provided with our text annotation method, our CNN-based predictive model generates state-of-the-art performance in churn prediction.

CCS Concepts
• Computing methodologies → Natural language processing

Keywords
Artificial intelligence (AI); convolutional neural networks (CNN); natural language processing (NLP); text mining; machine learning; churn prediction.

1. INTRODUCTION
For telecommunication service providers, a principal method for reducing costs and generating revenue is to focus on retaining existing subscribers rather than acquiring new customers. To support this strategy, it is important to understand customer concerns as early as possible to prevent churn: The customer action of canceling a subscription and moving to a competitive service provider.

Retaining existing customers for business growth is important because the cost of acquiring new customers is much higher than that of satisfying existing ones. When customers switch to another competitive service provider, it results in the immediate loss of business. Customers typically choose to switch providers when they are no longer satisfied with their existing service provider. The telecommunications industry views churn not as an instant or sudden action, but rather as the result of a long process through which the customer finally decides to leave after being unable to find a satisfactory solution. During that process, customers typically make phone calls to the provider's customer service representatives to express concerns and request a satisfactory solution. It is at this moment when churn signals appear in phone calls.

If service providers could detect these churn signals early on, and then figure out effective and personalized solutions to solve the individual concerns, there is a strong opportunity to retain customers, and therefore minimize business loss. Recognizing the signs of churn in the early stage is critical for any service provider that hopes to take effective steps to re-instill or improve customer satisfaction and retain the customer's business. Similar work on churn prediction has occurred based on social media data, such as microblog data, using either the traditional machine learning algorithms [1, 2] or CNN [3]. Although promising, these churn prediction results are still not satisfactory or their positive impact on business is still limited. Our analysis reveals that there are two main underlying factors that make social media data less than ideal for churn prediction modeling. The first factor is that customers usually begin by conducting phone calls to their service provider about their concerns rather than going to the social networks since service providers mainly use phone calls to answer customers' questions. In the future, the online chat will become more and more popular for customer service, and we can then combine the two kinds of data for churn prediction. As a result, phone call data at this moment is more reliable for early detection of churn signals. The second factor is that when training these predictive models, there is insufficient training data. For example, for AT&T, T-Mobile and Verizon, only 2299, 1073, and 2495 golden-truth examples, respectively, are used for training and testing [1, 2, 3]. So, the CNN model [3] with many parameters is not guaranteed to be trained well for churn prediction due to the curse of dimensionality.

For our prediction model experiment, we collaborate with one of the biggest telecommunication companies in the United States, to use the actual customers' phone call data for building a model of churn prediction that could, in turn, produce real impact on the telecommunication company's business. For this purpose, we use audio recordings of actual customer phone calls and apply speech recognition techniques to translate the audio data into text data (transcripts) for analysis. Given the massive amount of phone call transcripts and the rare event of churn signals, it is necessary to develop a scalable data-labeling method to provide sufficient

ICIAI 2019, March 15–18, 2019, Suzhou, China

ACM ISBN 978-1-4503-6128-6/19/03…$15.00
https://doi.org/10.1145/3319921.3319937

training data in a cost-affordable way for building the CNN-based predictive model. So, we developed a scalable data-labeling process we employed in order to build our model. Our churn prediction system consists of 4 core components. First, we collect the phone call transcript data from the database. We only collect the caller channel transcripts since our domain knowledge of telecommunication customer service interactions inform us that customers' churn signals are mainly available in the caller channel, not in the agent channel. Furthermore, the use of the caller channel can significantly reduce the amount of data for both data labeling and data analytics. Next, we label the ground-truth data as the training examples, using our domain knowledge about churns from the collected caller channel transcripts and the scalable data-labeling method. Then, we use natural language processing (NLP) algorithms for document tokenization and vector representation for tokens. The fourth and final component is training the text classification model for churn prediction using the CNN algorithm. Experiments show that when sufficient training examples are provided, our CNN model generates state-of-the-art performance for churn prediction according to the quantitative metrics of F1-score, precision and recall.

Our contributions are demonstrated in the following two aspects:

- We conduct the analysis for actual customers' phone calls for customer churn prediction, using only the caller channel transcripts from the caller-agent phone conversations, for churn prediction. This not only reduces the amount of data for both fast data labeling and data analysis, but also improves the prediction performance.

- We develop AI algorithms including NLP and CNN algorithms for churn prediction from transcripts of customer phone calls.

The rest of the paper is organized as follows. In Section 2, we discuss the research methodology in detail. In Section 3, we present the experimental results, and we conclude the paper with some discussions and the future direction of work in Section 4.

2. RESEARCH METHODOLOGY

There are four components involved in this research methodology for customer churn prediction in the telecommunication industry: Problem definition and text data preparation from phone calls, tokenization of the text data using NLP, embedded vector representation for tokens/words, and the supervised deep learning algorithms for churn prediction.

2.1 Problem Definition and Data Preparation

Our phone call data is from daily customer service phone calls of telecommunications companies. Each phone call consists of two channels: the agent channel and the caller channel. Each agent or caller channel consists of a sequence of utterances (usually a sentence in the transcript data). On average, the phone calls are about 45 minutes and their transcripts are long documents. According to our domain knowledge of telecommunications, the customer's churn information is mostly contained in the caller channel. Rarely is it found in the agent channel. As a result, we only collect the caller channel transcripts. This not only reduces the amount of data to review when labeling the data, but also has the potential to improve the prediction performance by discarding the irrelevant agent channel. We label the caller channel transcripts and use them as the training data. For ensuring the quality of the training data, we only label the caller channel transcripts that have call times longer than five minutes so there is sufficient text data.

2.2 Tokenization for Documents

Each caller channel transcript in the training examples are taken to be a document. In machine learning for text classification, documents need to be tokenized into individual words or tokens so that documents can be represented as feature vectors for training traditional machine learning models such as SVM and decision tree models, or each token can be represented as an embedded vector for deep learning algorithms. So, we first tokenize each document into a collection of terms. After some preprocessing for the tokens is conducted, we then provide an embedded vector representation for each word which is learned by the distributed representation learning algorithm word2Vec [4].

2.3 Embedded Vector Representation for Words

The traditional one-hot representation for words is not efficient for its high dimensionality, mutual orthogonality, and sparsity without capturing any semantic information from the words. The motivation of using the embedded dense vector representation is to reduce the dimensionality and, at the same time, provide semantic information for words. The word2Vec [4] is a distributed representation learning algorithm that captures the semantic information of individual words with the embedded dense vectors so that synonyms can be inferred from each other. It contains two related models, the continuous bag-of-words (CBOW) model, and the skip-gram model as shown in Figure 1~2, respectively.

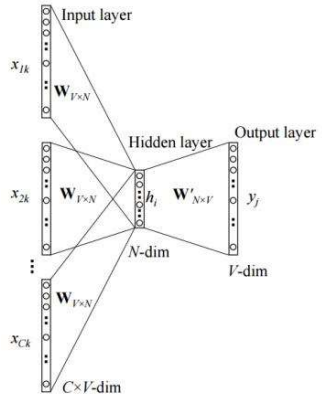

Figure 1. The CBOW model in word2Vec.

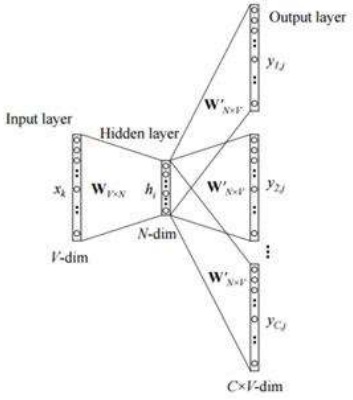

Figure 2. The Skip-gram model in word2Vec.

The CBOW predicts the current word from the context of its surrounding words in the sentence, within a window centered at the current word. On the other hand, the skip-gram model predicts the surrounding context words in a sentence from the current word, and the contexts are specified by a symmetric window centered at the current word. The Word2vec model can be trained with either the hierarchical softmax or negative sampling method in the iterative learning process of the two models. In the hierarchical softmax algorithm, it first creates a binary Huffman tree for all words in the vocabulary of the corpus according to their frequencies in the corpus. All words are the leaf nodes of the Huffman tree, and the high-frequency words have short paths from the root of the tree whereas the low-frequency words have long paths. When updating the embedded vector for a leaf node in the tree, it does not need all words' vectors to be involved in the calculation. Only the nodes (including both the leaf node and non-leaf-nodes) on the path from the root of the tree to the leaf node of the tree need to be involved in the calculation. As a result, the Huffman tree based hierarchical softmax algorithm can be used to optimize the original softmax algorithm by significantly reducing the computational complexity. On the other hand, for the negative sampling, it does not need all words' vectors to be involved in updating a word's vector in the backpropagation process by sampling a small number of the negative samples. Furthermore, for improving the vector quality of low-frequency words, the high-frequency words are down-sampled and the low-frequency words are up-sampled by the frequency lifting method because otherwise the high-frequency words are more likely to be sampled than the low-frequency words for updating their vectors. These models are the two-layer shallow neural networks. Word2vec takes as its input the high dimensional one-hot vectors of the words in the corpus and produces an embedded vector space of several hundred dimensions as specified such that each word in the corpus is represented by a unique low-dimensional dense vector in the embedded vector space. A very salient feature of this kind of word embeddings is that the vectors in the vector space are close to each other when their corresponding words are semantically close to each other.

2.4 Deep Learning for Churn Prediction

In this work, the supervised classification is adopted. To classify the documents into churns and non-churns, we have trained the convolutional neural networks (CNN) model since it outperforms the traditional machine learning algorithms as shown in the previous work for churn prediction [1, 2, 3].

CNN is one of the deep learning algorithms and it integrates the automatic feature extraction and feature selection together with the pattern classification algorithm in a single architecture. It has been widely used for text classification [5] and computer vision [6]. As outlined by Figure 3, for both computer vision and text classification, different channels of the data can be used as inputs of the CNN architecture for convolutional feature extraction. For computer vision applications, the R, G, B colors of an image are usually used as the inputs of the CNN. For text classification, the CNN algorithm usually takes as inputs the matrix of the word embeddings of the sentence. The embeddings can be either from word2Vec, one-hot representation or other vector representations of words, forming different channels of representation of the text data. Within the CNN architecture, each channel of the texts is represented as a matrix, in which, the rows of the matrix represent the sequence of words according to their order in a sentence, and each row is a word's vector representation. For feature extraction, the matrix is convolved with some filters of different sizes, respectively. All filters are with the same dimension as the words' embeddings.

The main idea of CNN for text classification with different filter sizes is pretty much like that for computer vision. The convolution operation is used to extract the semantic features (patterns) of different N-Grams characterized by the sizes of the filters. The different filter sizes correspond to different numbers of the N in N-Grams. The words' vectors can be either from the pre-trained word embeddings such as those of word2Vec from the large corpus, or randomly initialized. For the latter case, the word vectors are iteratively updated in the backpropagation process in the training stage until the model is learned. After the CNN model is trained, the word vectors become the side effects of the CNN model and this is another way of representation learning. Let's assume that the size of the filter w is m, the maximum length of sentence x is l, the dimensionality of word embeddings is d, then we have the sentence matrix $x \in R^{l \times d}$ and the filter $w \in R^{m \times d}$. During the convolution process, each of the filters gradually moves down one word at a time along the sequence of words. At each position, the filter covers a portion of the words' vector matrix, i.e., m rows of the sub-matrix, and the convolution is conducted. This convolution result is then squeezed by the rectified linear unit (Relu) activation function together with a bias term $b \in R$ to generate a feature value, which is mathematically represented in the following formula:

$$c_i = f(w \cdot x_{i:i+m-1} + b) \quad (1)$$

where the dot operation between matrix w and x is the element-wise multiplication operation, and the subscript i is the position index of the filter. After the convolution process is done for one filter, a list of feature values is obtained like $c = [c_1, c_2, c_3, \ldots, c_{l-m+1}]$, which is regarded as the feature map of the filter. Finally, the max-pooling operation continues to take the maximum value from the feature map as the output of the filter's convolution result with the sentence. When all filters are applied for convolution with the sentence's embedding matrix, we can get a list of feature values as the feature vector representation for the input sentence data. This feature extraction process with the max-pooling operation makes the length of the final feature vector independent of the input sentence length and the filter size. The length of the final feature vector is only dependent on the number of filters used for convolution. The final step in the CNN architecture is a full connection including the dropout strategy, regularization, and the Relu activation function from the final feature vector to the output layer. This full connection layer is fundamentally like the last layer of the conventional neural networks. The classification result of a sample is obtained by using the softmax function applied to the output layer for either binary classification or multi-class classification. The number of neurons in the output layer depends on the number of classes for the output.

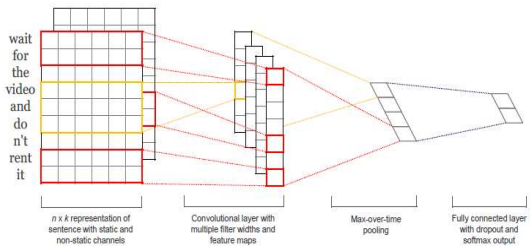

Figure 3. The CNN architecture for text classification with courtesy of Yoon Kim [3].

3. EXPERIMENTAL RESULTS AND ANALYSIS

To train the reliable predictive model for churn prediction, we annotate 20,000 samples of churny and non-churny caller channel transcripts as the training data and validation data.

The inputs of the words' vectors for the CNN are the pretrained word embeddings of word2Vec, and we have tried the embeddings of CBOW. For feature extraction through convolutions, we use 3 different filter sizes, 3, 4, and 5, and we use 128 filters for each filter size for feature extraction in the convolutional process. For the hyperparameters of CNN, our final settings are: batch size 32, epoch size 50, the dimension of word embeddings 300, dropout 0.5, decay coefficient 2.5, and $\boldsymbol{l_2}$-norm is not used for the regularization. The CNN algorithm is implemented in Python with the tools of Tensorflow and Numpy. In this paper, we use precision, recall, and F1-score as the quantitative metrics to measure the performance of the models, and they are calculated in the following way:

$$\boldsymbol{Precision} = \frac{\boldsymbol{tp}}{\boldsymbol{tp+fp}} * \boldsymbol{100} \quad (2)$$

$$\boldsymbol{Recall} = \frac{\boldsymbol{tp}}{\boldsymbol{tp+fn}} * 100 \quad (3)$$

$$\boldsymbol{F_1\_score} = \boldsymbol{2} \cdot \frac{\boldsymbol{precision * recall}}{\boldsymbol{precision+recall}} * \boldsymbol{100} \quad (4)$$

where *tp* denotes true positives, *fp* denotes false positives, and *fn* denotes false negatives. These metrics are multiplied by 100 in this paper for clarity consideration.

In the previous work [3], it has been reported that the CNN model with the pre-trained word embeddings outperforms the traditional machine learning models such as the naïve Bayes and SVM [1, 2], for which the bag of word (BOW) based vector space representation is used for document representation. So, in this work, we do not train these machine learning models again. Table 1 lists the previous prediction results of the CNN model with 4 different word embeddings for churn prediction using the microblog social media data [3]. It is clear to see that the difference of prediction performance for using the word embeddings of word2Vec and GloVe is marginal. So, in this paper, we only use the CBOW as the word embeddings for training our CNN model for churn prediction with phone call transcript data. Table 2 lists the previous prediction results of CNN + different logic rules for distilling the knowledge into the neural networks [3]. It demonstrates that when the logic rules are added, better prediction performance can be achieved. Table 3 lists our predictive model's performance on 819 testing examples for phone call data analysis. Our model is the CNN with the CBOW word embeddings. By comparing Table 2 with Table 3, it is easy to see that our CNN-based predictive model outperforms the published best model with the gain of 10 in F1-score and precision, generating state-of-the-art churn prediction performance. Even though we do not use additional logic rules to distill the knowledge into the neural networks, we still get much better prediction performance. Our analysis reveals that our improvement of prediction performance mainly comes from the fact that we have much more training examples to train the CNN model. Our model is trained with 20,000 training examples, and this greatly reduces the risk of being overfitted for the CNN model. However, those models in Table 2 use less than one tenth of our training examples to train those CNN models. According to the curse of dimensionality, when the CNN model has so many parameters for learning from the training data, it is hard to say that the used training examples for training those models in Table 2 are sufficient, so those models may not be optimized. This confirms the saying that for deep learning, the more the data, the better the performance!

Table 1. The reported churn prediction results of CNN for microblog data analysis using different word embeddings [3].

Input Vector(s)	F1-Score
Random embeddings	77.13
CBOW	79.89
Skip-Gram	79.55
GloVe	80.67

Table 2. The reported churn prediction results of CNN + three logic rules for microblog data analysis with/without using word embeddings [3]

Models	F1-Score	Precision	Recall
CNN	77.13	75.36	79.00
CNN+pretrained	80.67	79.28	82.11
CNN+pretrained+ "but" rule	81.95	80.84	83.09
CNN+pretrained+ "switch from" rule	80.92	79.74	82.14
CNN+pretrained+ "switch to" rule	82.60	80.89	84.39
CNN+pretrained+ All the 3 rules	83.85	82.56	85.18

Table 3. The prediction results of the proposed CNN model for phone call transcript data analysis.

Model	F1-Score	Precision	Recall
CNN + pretrained CBOW	93	93	93

4. CONCLUSION AND FUTURE WORK

In this paper, we develop NLP and supervised CNN algorithms for customer churn prediction for telecommunication industry. Through comparative studies with the recently published results, our CNN model generates state-of-the-art churn prediction performance for telecommunication industry. In the next step, we are going to further investigate the prediction performance in two directions. The first one is to continue to use CNN, but we will stack it with more layers to extract more abstract features for churn prediction. The second one is to try the LSTM, GRU, bi-LSTM and other variants to see if better performance can be achieved.

ACKNOWLEDGMENTS

Our thanks to ACM SIGCHI for allowing us to modify templates they had developed. The authors appreciate the great help from Brian Craig and Jana Baker in the revision of the paper.

REFERENCES

[1] Hadi Amiri and Hal Daume III, 2015. Target-dependent churn classification in microblogs. AAAI (2015). 2361-2367.

[2] Hadi Amiri and Hal Daume III, 2016. Short text representation for detecting churn in microblogs. AAAI (2016). 2566-2572.

[3] Mourad Gridach, 2017. Churn identification in microblogs using convolutional neural networks with structured logical knowledge. In Proceedings of the 3rd Workshops on Noisy User-generated Text (2017). 21-30.

[4] Mikolov Tomas, Chen Kai, Corrado Greg, Dean Jeffrey. 2013. Efficient estimation of word representations in vector space. In Advances in Neural Information Processing Systems (2013). 3111-3119.

[5] Yoon Kim, 2014. Convolutional neural networks for sentence classification. In Proceedings of the Conference on Empirical Methods in Natural Language Processing (2014). 1746-1751.

[6] Alex Krizhevsky, Ilya Sutskever, Geoffery Hinton. 2012. ImageNet classification with deep convolutional neural networks (2012). In 25th NIPS'1, 1097-1105.