Short Text Representation for Detecting Churn in Microblogs

Hadi Amiri and **Hal Daumé III**
Computational Linguistics and Information Processing (CLIP) Lab
Institute for Advanced Computer Studies
University of Maryland
{hadi,hal}@umiacs.umd.edu

Abstract

Churn happens when a customer leaves a brand or stop using its services. Brands reduce their churn rates by identifying and retaining potential churners through customer retention campaigns. In this paper, we consider the problem of classifying micro-posts as churny or non-churny with respect to a given brand. Motivated by the recent success of recurrent neural networks (RNNs) in word representation, we propose to utilize RNNs to learn micro-post and churn indicator representations. We show that such representations improve the performance of churn detection in microblogs and lead to more accurate ranking of churny contents. Furthermore, in this research we show that state-of-the-art sentiment analysis approaches fail to identify churny contents. Experiments on Twitter data about three telco brands show the utility of our approach for this task.

Introduction

Retaining customers is an important challenge for all businesses. Banks, telecommunication companies, airlines and other businesses utilize customer *churn* or *attrition* rate as a key business metric. Potential churners of a brand are those customers who are at the high risk of leaving the brand and its services. Identifying such customers, the first step of retention campaigns (Huang, Kechadi, and Buckley 2012), is crucial as the costs associated with retaining an existing customer greatly exceed the costs of acquiring a new one. This problem has been extensively studied on Call-Record data (CRD) in the context of social graphs of telecommunication companies (Verbeke, Martens, and Baesens 2014; Karnstedt et al. 2010) and to a lesser extent on online gaming (Kawale, Pal, and Srivastava 2009), chat and forum communities (Karnstedt et al. 2011; Oentaryo et al. 2012), and more recently on microblogs (Amiri and Daume III 2015).

The task of churn detection can be defined as follows: Given a micro-post and a target brand, determine if the micro-post is churny or non-churny with respect to the brand. For example the following tweets are churny with respect to "*Brand*-1": "*my days with Brand*-1 *are numbered*", "*i am leaving Brand*-1 *for Brand*-2!" , "*Brand*-1: *will change carriers as soon as contract is up.*".

Churn detection is a challenging task because of the following reasons (Amiri and Daume III 2015): First, the task is target-dependent and therefore comparative micro-posts in which users compare several brands against each other introduce a challenge to this task. Second, simple language constituents such as prepositions affect accurate churn detection. For example, consider the prepositions "*to*" and "*from*" in "*switch to Brand*-1" vs. "*switch from Brand*-1". Third, negation has an important contextual effect on churn detection as it simply reverse the label of a micro-posts. Fourth, *churny keywords*[1] do not necessarily indicate churny contents. For example, the micro-posts "*Brand*-1 *to give Brand*-2 *customers up to $450 to switch*" or " *I need a little Brand*-1*'s #help b4 leaving the states*" contain the terms "*switch*" and "*leaving*" respectively but they are not churny with respect to the brands. Finally, churny contents can be expressed in subtle ways such as "*debating if I should stay with Brand*-1" or "*in 2 months, bye Brand*-1" that contain no obvious churny keywords.

One may suspect that an state-of-the-art sentiment analysis system can be used to identify churny contents. Therefore, it might be enough to rely on such a system and consider negative micro-posts about a target brand as churny and positive or neutral ones as non-churny. However, this approach fails because of the following three reasons: (a) positive micro-posts could be churny, e.g. the tweet "*hate that i might end up leaving Brand*-1 *cuss they are the best company ever*" is churny even though it expresses positive opinion about the brand, (b) neutral micro-posts could be churny as well, e.g. the tweet "*I am leaving Brand-1 on March 7th*" is churny while it's of neutral orientation as it does not explicitly express any positive or negative opinion about the brand, and (c) negative micro-posts could be non-churny, e.g. the tweet "*Brand-1 cell coverage still sucks*" expresses a negative opinion about the brand while it's non-churny as it does not indicate that the user is leaving the brand [2]. In fact, our experiments shows that state-of-the-art sentiment classifiers perform poorly on churn classification

[1] such as: {*leave, leaving, switch, switching, numbered, cancel, canceling, discontinue, give up, call off, through with, get rid, end contract, break up,* , . . . }.

[2] Note that, although churn and sentiment detection are different tasks, sentiment signals could be useful for detecting churny contents as such contents may carry negative sentiment toward brands.

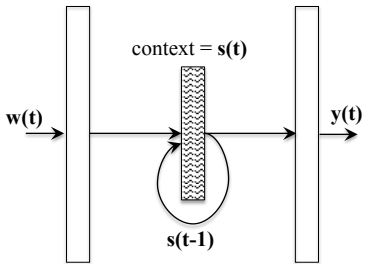

Figure 1: RNN for tweet representation. The recurrent connection at the hidden layer allows information about previously seen words to cycle inside the network and be memorized.

(see Table 1 in experiments). This result indicates that churn and sentiment are fundamentally different concepts and new techniques and approaches need to be developed for effective churn detection.

Motivated by the recent success of recurrent neural networks (RNNs) in word representation (Bengio et al. 2003; Mikolov et al. 2013), we propose to utilize RNNs to tackle the above problem by learning micro-post representation. We show the utility of such representations in target-dependent churn detection in microblogs. Experiments on a Twitter dataset created for three telecommunication brands show an average F1 performance of 78.30% for target-dependent churn classification in microblogs.

Micropost Representation

Artificial neural networks have been found effective in learning vectors to represent words and their similarities. In this Section, we propose to utilize a simple recurrent neural network (RNN) model, called Elman network (Elman 1990), to represent micro-posts in such a vector space. This approach can be used to encode tweets with arbitrary lengths into vectors of fixed length through recurrent connections. The resultant vectors can then be used as features for classification, clustering, topic modeling, etc.

The Elman RNN model is shown in Figure 1. It has an input layer $\mathbf{x} \in \mathbb{R}^{v+m}$, a hidden layer $\mathbf{s} \in \mathbb{R}^m$ (also called context layer), and an output layer $\mathbf{y} \in \mathbb{R}^v$ where v is the size of vocabulary and m is the size of the hidden layer. Let $\mathbf{x}(t)$, $\mathbf{s}(t)$, and $\mathbf{y}(t)$ represent the input to the network, the state of the network, and the output of the network at time t respectively. Our network is provided by sequences of words where words (along with other potentially useful information) are given one by one to the network at each time. In particular, at each time t, we create the input vector $\mathbf{x}(t) \in \mathbb{R}^{v+m}$ by concatenating the vector of the input / current word $\mathbf{w} \in \mathbb{R}^v$ and the context layer vector at time $t-1$, $\mathbf{s}(t-1) \in \mathbb{R}^m$ (see our discussion below of why the context layer provide useful information). Note that, initially, $\mathbf{w}_i = 1$ where i is the index of the current word in our vocabulary and $\mathbf{w}_j = 0$ for all $j \neq i$.

The recurrent connection at the hidden layer allows information about previously seen words to cycle inside the network. In fact, the hidden layer $\mathbf{s}(t)$ can be thought of as a memory that remembers / represents the words that have been observed up to time t. The model is independent of the length of the input as by the time that we processed the last word of a tweet, the context layer, $\mathbf{s}(.)$, can be used to represent the entire tweet in the vector space.

Formally, the input, output, and hidden layers can be computed as follows:

$$\mathbf{x}(t) = [\mathbf{w}(t), \mathbf{s}(t-1)] \tag{1}$$

$$\mathbf{s}(t) = f(\mathbf{U}\mathbf{x}(t)) \tag{2}$$

$$\mathbf{y}(t) = g(\mathbf{V}\mathbf{s}(t)) \tag{3}$$

where $\mathbf{U} \in \mathbb{R}^{m \times (v+m)}$ and $\mathbf{V} \in \mathbb{R}^{v \times m}$ are the weight matrices to learn, $f(.)$ is the sigmoid activation function:

$$f(z) = \frac{1}{1+e^{-z}} \tag{4}$$

and $g(.)$ is the softmax function:

$$g(z_m) = \frac{e^{z_m}}{\sum_k e^{z_k}}. \tag{5}$$

The above Equations are used to learn the weights and activation of the hidden and output layers. The sigmoid function adds non-linearity to the model, while, assuming that the output layer represents probability distribution of next word given $\mathbf{w}(t)$ and context $\mathbf{s}(t-1)$, the softmax function ensures that the probability distributions are valid, i.e. $\mathbf{y}_m(t) > 0$ and $\sum_k \mathbf{y}_k(t) = 1$. The network is trained by backpropagation through time (BPTT) (Rumelhart, Hinton, and Williams 1988) in witch all the training data is sequentially presented.

Finally, once the model is trained, for each input tweet d_i with n_i words, we use the context layer vector at $t = n_i$, i.e. $\mathbf{s}(t = n_i)$, to represent the tweet (we refer to this model as ***tRep*** in experiments).

Furthermore, another approach to represent a tweet is to compute the average word embedding for all words that appear in the tweet (we refer to this approach as ***wRep***). This average vector can be considered as a basic way to represent the entire tweet and its performance can be compared against our RNN tweet representation approach above, ***tRep***. Furthermore, we can strengthen these representations by concatenating them (this concatenation is referred to as ***twRep*** in experiments).

Representation of Churn Indicators

Amiri and Daume III (2015) introduced three categories of churn indicators: *demographic*, *content*, and *context* churn indicators. Demographic indicators / features are extracted from user profiles, while content and context indicators are extracted from the content of micro-posts and discussion threads respectively. Since the focus of our work is on micro-post and content representation, we only utilize and extend the content and context churn indicators in this research.

Representation of Content Churn Indicators

Words of an input tweet can be readily used as classification features. As discussed above, we can also utilize vector representation of such words, *wRep*, *tRep*, and *twRep* respectively for classification.

In addition to bag of words features, the neighboring words of target brands in micro-posts often contain rich contextual information for churn classification. We capture the effect of neighboring words of brands / competitors by considering $2k$ features representing the k left and the k right neighboring words of the brand, and $2k$ neighboring-word features for competitors (we set $k = 3$ as it leads to superior performance in the experiments). We can also utilize the embedding vectors of neighboring words as classification features. Note that, since k is given, we can concatenate the $2k$ embedding vectors instead of averaging them. Indeed this concatenation greatly improves the performance as compared to simple averaging. We refer to this model as ***NbRep*** in experiments.

Dependency Path We propose to effectively utilize syntactic relations between words to identify expressions that describe target brands. In particular, we introduce the concept of *dependency path* to find such expressions and their corresponding RNN representations. We define a dependency path in a dependency tree as a sub-tree that covers the path from the root of the tree to the target brand node and all its children. Figure 2 shows a dependency path example for the target "*Brand*-1". A dependency path contains all the words in an input micro-post that have some direct / indirect syntactic relations with the target brand. We utilize words on dependency paths and their corresponding representations (see below) as classification features. Furthermore, to capture the negation effect, if a word on a dependency path is negated, we include the corresponding negation dependency to the dependency path.

We create RNN dependency features by computing a vector representation for each dependency path. For this purpose, we utilize our RNN model described in previous Section. For example, given the tweet "*i want to switch from crappy Brand*-1 *to Brand*-2 *or Brand*-3", we utilize our RNN model to compute a representation vector for the input "*want switch from crappy Brand*-1" obtained from the dependency path (note that it is important to consider words based on the order that they appear in the original tweet). This approach can help identifying important features that directly affect the target brand while ignoring other less important features. We refer to this model as ***DepRep*** in experiments. We note that another method to combine such representations is to average the embedding vectors of words that appear on the dependency path. However, our experiments show that the first approach is significantly more effective. In the future, we aim to utilize the types of such dependency relations for tweet representation.

Representation of Context Churn Indicators

Churny tweets may trigger discussions between users, their friends, and brand representatives on social media platforms.

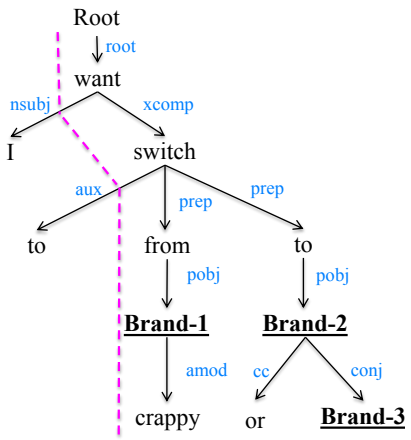

Figure 2: The subtree extracted for the target "Brand-1". The red line shows the *dependency path* for the target brand. We can represent the entire tweet by computing a representation vector for the input "*want switch from crappy Brand*-1" obtained from the dependency path. Note that words are considered based on the order that they appear in the original tweet.

Brands participate in such discussions to retain their customers while competitors intervene to hunt new customers.

To capture useful information in such discussion threads, we extract content features (as discussed in the above Section) for each micro-post in the thread. To distinguish the content generated by different parties in threads, in the bag of words model we put the features into different namespaces by adding "*USer*-", "*FRiend*-", "*BRand*-", and "*COmp*-" prefixes to the content features extracted from user, friend, target brand, and competitor microposts respectively (we refer to this model as ***Cntx*** in experiments). In our RNN model, we compute four representation vectors for the contents generated by users, friends, target brands, and their competitors in threads respectively (we refer to the concatenation of these four vectors as ***CntxRep*** in experiments).

We note that there exists other churn indicators in threads. For example, the number of brand or competitors posts in the thread or the reciprocity between user and brand posts are good indicators. However, we only utilize content features and their representations from threads for fair comparison.

Churn Data

We utilize churn data provided by (Amiri and Daume III 2015)[3]. The data was collected from twitter for three telecommunication brands: *Verizon*, *T-Mobile*, and *AT&T*. The high inter-annotator agreement reported, Fleiss' kappa (Fleiss 1971): 0.62, and Cohen's kappa (Cohen 1960): 0.93, indicates that the task is fairly feasible for human annotators (note that Fleiss κ and Cohen κ values are not directly comparable).

[3] www.umiacs.umd.edu/~hadi/chData/

	Verizon			AT&T			T-Mobile		
	P$^+$	**R$^+$**	**F1$^+$**	**P$^+$**	**R$^+$**	**F1$^+$**	**P$^+$**	**R$^+$**	**F1$^+$**
MetaMind	30.52	42.28	35.45	15.47	46.59	23.23	25.46	36.56	30.02
NRC	29.25	54.45	38.06	12.06	69.32	20.54	28.51	68.54	40.27
SemEvalSnt	28.49	54.76	37.48	11.86	65.91	20.10	28.58	68.37	40.31
TargetSnt	29.86	60.84	40.06	14.16	60.12	23.51	30.37	63.20	41.03
Unigram	58.59	73.41	65.17	60.46	73.91	66.32	58.66	70.50	64.04
Unigram+Nb	**70.51**	**77.70**	**73.70**	**79.29**	**77.41**	**77.92**	**67.39**	**71.00**	**69.15**

Table 1: F1 Performance of baselines for churn classification (reported for the churn class).

Furthermore, we utilize a large dataset that contains more than 6M tweets about the above brands as development dataset. We note that, naturally, churny contents occur less frequently than non-churny contents and as such we believe our development dataset is highly imbalanced in terms of churny and non-churny tweets. We will further discuss the effect of balanced and imbalanced data in experiments.

Experimental Results

We employ two state-of-the-art classification approaches for churn classification in microblogs. In particular, we consider the hinge loss (employed by the SVMs) and the logistic loss (representing logistic classifier). We employ Vowpal Wabbit classification toolkit[4] with all parameters set to their default values to perform the classification experiments.

In the experiments, we report classification performance (F1-score) over the churn class. We performed all the experiments through 10-fold cross validation and used the two-tailed paired t-test $\rho < 0.01$ for significance testing. Here, we use the asterisk mark (*) to indicate significant improvement over baselines.

We use our development dataset to learn our RNN model for tweet representation. For this, we set the size of hidden layer to $m = 128$ in the experiments. We note that greater values of m may lead to higher performance.

Performance of Baselines

We study the performance of the state-of-the-art sentiment classification approaches as well as n-gram models to determine a strong baseline for this task. For sentiment baselines, we treat positive or neutral tweets about a brand as non-churny and negative tweets as churny. We consider the following baselines in this study:

Baseline-1 (MetaMind): We consider the twitter sentiment classifier of MetaMind as our first baseline [5]. This classifier is an accurate sentiment classifier that is trained over 331K general tweets and has an accuracy of 81.73% for sentiment classification.

Baseline-2 (NRC) (Mohammad, Kiritchenko, and Zhu 2013): NRC is the top-ranked Twitter sentiment classifier of SemEval 2013. It deploys an SVM classifier with linear kernel in conjunction with a wide range of features (see details below).

[4] http://hunch.net/~vw/

[5] https://www.metamind.io/classifiers/155

Baseline-3 (SemEvalSen) This baseline is an ensemble learning approach that combines several state-of-the-art Twitter sentiment classification approaches that participated in the "Sentiment Analysis in Twitter" task of SemEval 2015. These approaches are NRC-Canada (Mohammad, Kiritchenko, and Zhu 2013) (baseline-2), GU-MLT-LT (Günther and Furrer 2013), and KLUE (Proisl et al. 2013). They utilize various features for effective sentiment analysis on tweets including n-grams, sentiment lexicons, punctuations, emoticons and abbreviations, word lengthening, tweet clustering, and techniques for handling negations. The approaches are combined in an ensemble by averaging the individual classifier confidence scores to determine the resultant sentiment of tweets (Hagen et al. 2015).

Baseline-4 (TargetSen): The above approaches are target-independent sentiment classifiers and as such may assign irrelevant sentiment to tweets and target brands (especially for comparative micro-posts in which several brands are compared against each other). As our fourth sentiment baseline, we consider the approach proposed by Jiang et al. (2011) for target-dependent Twitter sentiment classification. This approach utilizes most of the aforementioned features as well as target-dependent features obtained from dependency relations between words and target brands in tweets. Furthermore, this approach incorporates similar tweets for more effective sentiment classification with respect to a given target (not utilized in this paper).

Baseline-5 (unigram+Nb): We experimented with different ngrams (n={1,2,3}) and their combination at the word and character levels (Amiri and Daume III 2015). We found that the combination of unigrams and neighboring words (see Section 3) leads to the best performance. Thus, we consider this setting as a strong baseline for this task (and refer to it as unigram+Nb).

The performance of the above approaches are reported in Table 1 respectively. The results clearly indicate that sentiment classifiers are not effective for churn classification. As discussed before, we attribute this result to the fact that objective contents could potentially be churny while subjective contents might be non-churny. Sentiment classifiers fail to capture such differences.

Performance of RNN Representation

Tables 2 shows the Macro-average F1 Performance for Bag of Words (BOW) and RNN models evaluated on all three brands, Verizon, AT&T, and T-Mobile. The rows show the performance of corresponding features for the two models

	BOW			RNN		
	BOW Features	**hinge**	**logistic**	**hinge**	**logistic**	**RNN Features**
(1)	unigram	65.30	64.30	62.97	63.73	tRep
				59.97	61.37	wRep
				66.13*	**66.20***	twRep
(2)	unigram+Nb	73.63	72.17	71.07	**73.90***	twRep+NbRep
(3)	unigram+Dep	72.40	71.80	**75.66***	75.43*	twRep+DepRep
(4)	unigram+Cntx	74.27	73.20	**75.47***	75.03*	twRep+CntxRep
(5)	unigram+Nb+Dep+Cntx	77.03	75.60	76.77	**77.56***	twRep+NbRep+DepRep+CntxRep
(6)	**BOW+RNN:** hinge: **78.30**, logistic: 78.15					

Table 2: Macro-average F1 Performance for all three brands. The performance is reported for the churn class. The average F1 improvement in the BOW model is 3.3% and the corresponding improvement for the RNN model is 4.68% as compared to the baseline (unigram+Nb and *twRep+NbRep* respectively). Significance test (indicated by *) indicate the cases where our RNN model significantly outperforms the BOW model for the same classification approach and configuration row.

(presented on the left and right sides of the Table respectively). In Tables 2, significance test reports (shown by *) indicate the cases where our RNN model significantly outperforms the BOW model for the same classification approach and configuration row.

Row (1) shows the performance of unigram model and our RNN-based representation models. Recall that *tRep*, *wRep*, and *twRep* show the performance when we use context layer vectors, average word embeddings, and their concatenation for tweet representation respectively. As the results show, *tRep* produces a comparable performance to unigrams while it only uses a feature space of length $m = 128$ (as compared to the length of vocabulary in case of unigrams) for classification. The concatenation of *tRep* and *wRep*, i.e. *twRep*, leads to significantly higher performance than unigrams.

We expected greater improvement by our RNN model. The small improvement is because of the fact that our RNN model is trained over a heavily unbalanced (in terms of churny and non-churny tweets) development dataset. We used an unbalanced development dataset to evaluate the performance of our model in real world situations. Even with such an unbalanced development dataset, the *twRep* model significantly outperform unigrams across all the three brands. In the next Section, we show that a more balanced development dataset will improve the representation power of our RNN models.

Row (2) shows that the combination of unigrams and neighboring words greatly improves the performance in both BOW and RNN models. This improvement is because neighboring word features capture word orders and provide more context information. In case of RNN representations, we observed higher performance by concatenating the neighboring word vectors instead of averaging them. This result and the performance of *wRep* suggests that averaging word vectors (in contrast to concatenating them) may lead to information loss and inversely affect the performance.

Row (3) shows that dependency relations improve the performance over unigrams with greater improvement in the RNN model. The higher improvement in case of RNN representations is due to vector representation through dependency paths. In fact, as discussed before, dependency paths help to extract important features that directly affect the target brand and ignore other less important features.

Row (4) shows that adding context indicators (***Cntx*** and ***RepCntx*** respectively) to the unigram features and RNN representations improves the performance and outperform the unigram+Nb and *twRep+NbRep* models respectively. This indicates that discussion threads provide useful information for churn detection.

Row (5) shows that combining all content and context features in each category improve the performance over the baseline in both models. The average improvement in the BOW model is 3.3% F1 score and the corresponding improvement for the RNN model is 4.68% as compared to their corresponding baselines (unigram+Nb and *twRep+NbRep* respectively). Finally, row (6) shows that the combination of BOW and RNN representation models leads to the best performance.

Performance of RNNs with Balanced Data

In this Section, we evaluate the effect of a balanced development dataset on the performance of our RNN model for churn detection. To obtain a development dataset that is more balanced, we first utilized the best performing classifier above (SVMs, see row (6) in Table 2) to label each tweet in our large development dataset as churny or non-churny. Based on the classification results, the ratio of churny vs. non-churny tweets in our original development dataset is 0.025 that indicates our original development dataset is highly imbalanced. Then, we created a more balanced development dataset by considering all tweet that were labeled as churny in conjunction with an equal number of randomly selected tweets that were labeled as non-churny. This dataset contains more than 300K tweets.

We repeated our experiments with this new development dataset. Table 3 shows the results: our *tRep* model alone outperforms the unigram model using more balanced development dataset. As Table 3 shows this consideration also improves the performance of *wRep* and *twRep* models respectively. This improvement is because of the more balanced development dataset that allows our RNN model to learn more accurate representation of churny micro-posts.

	RNN Features	hinge	logistic
(1)	tRep	65.93	67.23
(2)	wRep	60.47	62.01
(3)	twRep	67.77	68.27
(4)	twRep+NbRep	74.23	75.89

Table 3: Macro-average F1 Performance for all three brands with balanced development data.

Ranking Performance

Marketing departments of brands would often want to investigate the most churny tweets first. As such, we propose to evaluate the performance of churn detection approaches in terms of their ranking of churny tweets. We expect a good churn detection model to rank the most churny tweets higher in its ranking list of churny tweets. For this evaluation, given the output of a classification model, we rank tweets that have been labeled as churny based on the predicted values of our classifiers (i.e. probability scores produced by the logistic classifier or confidence values predicted by SVMs). We can then plot an interpolated precision-recall diagram (Baeza-Yates and Ribeiro-Neto 1999) to evaluate the quality of different rankings. Figure 3 shows the results for *unigram+Nb* and *twRep+NbRep* churn classification models averaged over all the three brands. As the results show, at higher ranks (i.e. lower recall values) *twRep+NbRep* has greater precision than *unigram+Nb*. This indicates that the *twRep+NbRep* model ranks true churny tweets higher in the list and as such produces a better ranking of churny tweets as compared to the baseline, *unigram+Nb*.

Related Work

The problem of churn classification has been studied on Call-Record data (CRD) where the input is a social graph in which nodes are customers and edges show communications / telephone calls between nodes (Verbeke, Martens, and Baesens 2014; Huang, Kechadi, and Buckley 2012; Karnstedt et al. 2010). Various indicators have been shown to be important for effective churn detection on CRD. These include node-specific information such as user age, gender, bill frequency, account balance, outstanding charges and historical information of bills and payments, as well as network-specific features including information about calls, such as call duration and prices, and information about incoming and outgoing calls and the parties involved.

Churn detection has also been studied in the context of online gaming (Kawale, Pal, and Srivastava 2009) and chat and forum communities (Dror et al. 2012; Karnstedt et al. 2011; Oentaryo et al. 2012; Patil et al. 2013) where the purpose is to detect if a user stops playing a game or leaves a chat forum. Churn classification in these domains also utilized various user-specific features such as age, gender, and rate of activity, as well as features obtained from social relations between people and their interactions.

Recently Dave et al. (2013) presented a churn classifier for recommender systems in which they aimed to predict if a user returns to use a target product after his / her initial experience with the product. The approach utilizes user-specific features such as user ratings and the time that users spend on items to predict churn.

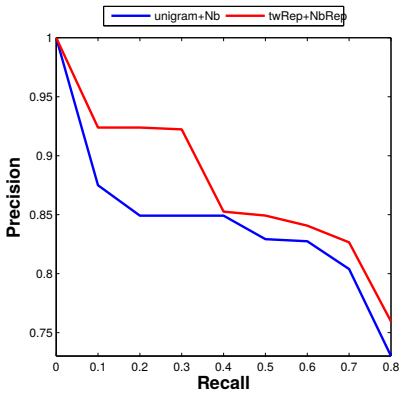

Figure 3: Interpolated Precision-Recall for a ranked list of churny tweets. Note that interpolated precision at recall r is defined as the highest precision found for any recall $r' \geq r$.

Despite the numerous applications of user retention and the availability of public user generated contents in microblogs, target-specific churn classification appears to be under-explored in microblogs. In fact, microblog platforms provide information about different aspects of brands that can be effectively utilized to identify potential churners. Recently Amiri and Daume III (2015) studied the problem of churn detection in microblogs in which they aimed to utilize user generated contents to predict churn. As discussed here, they introduced several important churn indicators.

In this work, we focused on the language modeling and evaluation aspects of churn in microblogs and presented models for tweet and churn indicator representation. Furthermore, we showed that the task is fundamentally different from sentiment analysis and as such new techniques and methods need to be developed for target-dependent churn detection in microblogs.

Conclusion and Future Work

We investigated the problem of target-dependent churn classification in microblogs. We proposed a simple recurrent neural network to represent tweets through fixed-length vectors and utilized the resultant representations for churn classification. Furthermore, we examined and extended factors that make churn detection in microblogs more accurate.

As our future work, we aim to further investigate this problem from the content representation perspective. For example, consider the churny tweet "*Brand*-1 *great new #moreofnothing plans! can't wait until match 7th to switch.*" This tweet not only indicates a potential churn but also its time frame. We aim to extend our models to predict the time that churn will be happening. Furthermore, we observed that syntactic relations between words are highly important in capturing the complexity of churny language. In the future, we aim to develop techniques to utilize the types of dependency relations to capture such complex effects.

References

Amiri, H., and Daume III, H. 2015. Target-dependent churn classification in microblogs. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*.

Baeza-Yates, R., and Ribeiro-Neto, B. 1999. *Modern information retrieval*, volume 463. ACM press New York.

Bengio, Y.; Ducharme, R.; Vincent, P.; and Janvin, C. 2003. A neural probabilistic language model. *J. Mach. Learn. Res.* 3:1137–1155.

Cohen, J. 1960. A coefficient of agreement for nominal scales. *Educational and psychological measurement* 20(1):37–46.

Dave, K. S.; Vaingankar, V.; Kolar, S.; and Varma, V. 2013. Timespent based models for predicting user retention. In *Proceedings of the 22nd international conference on World Wide Web*, 331–342. International World Wide Web Conferences Steering Committee.

Dror, G.; Pelleg, D.; Rokhlenko, O.; and Szpektor, I. 2012. Churn prediction in new users of yahoo! answers. In *Proceedings of the 21st international conference companion on World Wide Web*, 829–834. ACM.

Elman, J. L. 1990. Finding structure in time. *COGNITIVE SCIENCE* 14(2):179–211.

Fleiss, J. 1971. Measuring nominal scale agreement among many raters. *Psychological Bulletin* 76(5):378–382.

Günther, T., and Furrer, L. 2013. Gu-mlt-lt: Sentiment analysis of short messages using linguistic features and stochastic gradient descent.

Hagen, M.; Potthast, M.; Büchner, M.; and Stein, B. 2015. Webis: An ensemble for twitter sentiment detection. In *Proceedings of the Ninth International Workshop on Semantic Evaluation (SemEval)*.

Huang, B.; Kechadi, M. T.; and Buckley, B. 2012. Customer churn prediction in telecommunications. *Expert Syst. Appl.* 39(1):1414–1425.

Jiang, L.; Yu, M.; Zhou, M.; Liu, X.; and Zhao, T. 2011. Target-dependent twitter sentiment classification. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT 2011, 151–160. Stroudsburg, PA, USA: Association for Computational Linguistics.

Karnstedt, M.; Hennessy, T.; Chan, J.; Basuchowdhuri, P.; Hayes, C.; and Strufe, T. 2010. Churn in social networks. In Furht, B., ed., *Handbook of Social Network Technologies and Applications*. Springer US.

Karnstedt, M.; Rowe, M.; Chan, J.; Alani, H.; and Hayes, C. 2011. The effect of user features on churn in social networks. In *Proceedings of Web Science*. ACM.

Kawale, J.; Pal, A.; and Srivastava, J. 2009. Churn prediction in mmorpgs: A social influence based approach. In *Proceedings of International Conference on Computational Science and Engineering*. IEEE Computer Society.

Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G. S.; and Dean, J. 2013. Distributed representations of words and phrases and their compositionality. In Burges, C.; Bottou, L.; Welling, M.; Ghahramani, Z.; and Weinberger, K., eds., *Advances in Neural Information Processing Systems 26*. Curran Associates, Inc. 3111–3119.

Mohammad, S. M.; Kiritchenko, S.; and Zhu, X. 2013. Nrc-canada: Building the state-of-the-art in sentiment analysis of tweets. In *Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval)*.

Oentaryo, R. J.; Lim, E.-P.; Lo, D.; Zhu, F.; and Prasetyo, P. K. 2012. Collective churn prediction in social network. In *Proceedings International Conference on Advances in Social Networks Analysis and Mining*.

Patil, A.; Liu, J.; Shen, J.; Brdiczka, O.; Gao, J.; and Hanley, J. 2013. Modeling attrition in organizations from email communication. In *Social Computing (SocialCom)*.

Proisl, T.; Greiner, P.; Evert, S.; and Kabashi, B. 2013. Klue: Simple and robust methods for polarity classification. In *Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval)*.

Rumelhart, D. E.; Hinton, G. E.; and Williams, R. J. 1988. Neurocomputing: Foundations of research. Cambridge, MA, USA: MIT Press. chapter Learning Representations by Back-propagating Errors, 696–699.

Verbeke, W.; Martens, D.; and Baesens, B. 2014. Social network analysis for customer churn prediction. *Applied Soft Computing* 14,C(0):431–446.