Finding Related Tweets on Twitter: A Statistical Relational Learning Approach

Navdeep Kaur
navdkaur@umail.iu.edu

Abstract

In this project we employ a successful SRL model called RDN-Boost to predict if two tweets are related with each other or not and by what certainty. We use the crawled data from twitter for learning the relation between tweets and perform a set of experiments to evaluate the performance of learnt model on the test dataset. Our results demonstrate that RDN-Boost is a promising model for extracting the relations in the social media data. In addition to producing excellent results, RDN-Boost is also able to reason about why two tweets are related by producing relation trees as output.

Introduction

Twitter data contains rich information on a vast scale that can be utilized for a variety of research purposes. This has naturally drawn the attention of machine learning community where machine learning algorithms have been applied to Twitter data for the applications like sentiment analysis (Severyn and Moschitti 2015), discovering similar users (Goel et al. 2015), and paraphrase identification (Xu, Callison-Burch, and Dolan 2015) etc.

Though successful, standard machine learning algorithms applied on Twitter data assume data to be *fixed-size flat feature vector* whereas in real world, each data point might have different number of features. For instance, if tweet words are used as input features, two tweets can have different number of words. Further, conventional machine learning assumes data points to be independent of each other (*iid assumption*), whereas real-world data is inherently relational i.e. they consist of inter-related objects and dependencies. For e.g. two twitter users are related because they follow each other. Representing relational data as a *flat feature vector* results in loss of information (Jensen and Neville 2002). Thirdly, there is *noise* and *uncertainty* in real data that should be handled by a machine learning model. for e.g. 'thanks' and 'thnx' refer to same word in twitter slang.

The branch of machine learning that can handle uncertainty and relations present in real world data during learning is called Statistical Relational Learning (SRL) (Getoor and Taskar 2007). SRL models integrate either first order logic or relations with probability which results in interpretable models. Though plethora of SRL models have been proposed in the past decade, one such state-of-the-art model SRL model we are interested in is RDN-Boost (Natarajan et al. 2012) that can combine a set of weak models during learning and has successfully predicted the probabilistic relations between two object (tweets in this project). In addition, RDN-Boost provides relational trees that can reason about the presence of relations between objects.

To show the applicability of SRL models on twitter data, we formulate the following machine learning problem in this project: *(i)* the aim of this project is to predict if two tweets are related with each other or not, if yes, then by what certainty *(ii)* we consider text of tweet as main feature of the data to be fed to SRL model for prediction. In addition to the words in the tweet, we also consider emoticons, emojis, hyperlinks etc present in text as features of the model in order to see how important they are to predict relations in tweets *(iii)* we apply an SRL model namely RDN-Boost on Twitter data to predict if two tweet are related to each other *(iv)* we assume that two tweets are related if they share a hashtag because two people are talking about the same topic mentioned in the hashtag. This assumption is used as gold standard to label the data.

The rest of the paper is organized as follows: First, we perform a detailed literature survey of past work that can predict the relation between tweets. We next discuss RDN-Boost model in detail. We further discuss how relational data was collected from Twitter. We next present how RDN-Boost perform on the challenging task of finding relations in twitter data. Finally, we conclude the paper by discussing the limitations of the project and area of further research.

Literature Survey

The past research on finding related tweets can broadly be categorized into two categories: *embedding based approaches*, and *non-embedding based approaches*. An embedding (Mikolov et al. 2013) is a vector representation of word or sentence such that two semantically similar words will lie close to each other in feature space. Under embedding based methods, two sub-categories exist: models that predict the hashtag of the text, given the tweet as input (*topical similarity*) and models that can predict *semantic similarity* between tweets without hashtags.

(Weston, Chopra, and Adams 2014)'s *Tagspace* model operates at the word level where pretrained word-embedding

Table 1: Comparison of Machine Learning Models for finding similar tweets. GRU: Gated Recurrent Unit; NN: Neural Network; Topical: Two tweets are similar because they share hashtag; S: Supervised Model; U: Unsupervised Model

Model	**Base Model**	**Supervised**	**Embeddings based**	**Relational Aspect considered during learning**	**Similarity Type**
(Dhingra et al. 2016)	Bi-directional GRU	S	yes	No	Topical + Semantic
(Weston, Chopra, and Adams 2014)	Convolutional Neural Network	S	yes	No	Topical + Semantic
(Le and Mikolov 2014)	NN (linear hidden layer)	U	yes	No	Semantic
(Guo, Liu, and Diab 2014)	Matrix Factorization and HashMaps	U	yes	no	Topical
(Godin et al. 2013)	Latent Dirichlet Allocation	U	No	No	Topical
(Zarrella et al. 2015)	ensemble of ML Models	U + S	Yes, for some models	No	Semantic
(Natarajan et al. 2012)	ensemble of trees	S	no	**Yes**	Topical

of words in a tweet are input to convolutional neural network and hashtags to be predicted acts as a label at the output layer. The method can rank most suitable hashtags for a tweet which is empirically proved as it outperforms baselines. On the other hand, (Dhingra et al. 2016)'s *tweet2vec* model operates at character level granularity, where all the characters in a tweet are input to Bi-directed Recurrent Unit (Bi-GRU) that produces a probabilistic score of a tweet belonging to a hashtag. The experimental data consists of 2 million tweets crawled from twitter and this model outperforms word-level granular models proving the superiority of considering character-level inputs. Even though above models predict hashtag of one tweet, we can infer that two tweets are related if they have same hashtag.

The above models find the topic (hashtag) similarity between tweets. There is other line of research under embedding based models that find semantic similarity between tweets without a given hashtag: For instance, (Zarrella et al. 2015) considered ensemble of models to find paraphrase and semantic similarity between two tweets; where each model in ensemble produces similarity score between tweets and individual models in ensemble range from recurrent neural network to string similarity based models. This method was among the top performers in SemEval-2015 Task 1.

(Le and Mikolov 2014) learns a neural network model by considering words contained in a paragraph and results in paragraph embeddings that incorporate the semantic meaning of entire paragraph. The method outperforms state-of-the-art methods on (i) sentiment analysis performed on Stanford Sentiment Treebank and IMDB datasets (ii) information retrieval task. Though it's not explained in the paper, its trivial to find the similarity between two paragraphs (tweets) by calculating cosine similarity between them.

In addition to embedding based models discussed above, *non-embeddings* models of hashtag prediction also exists. For instance, (Godin et al. 2013) proposes two step learning method: (i) a Naive Bayes based EM method that can distinguish between English and non-English tweets. (ii) once English tweets can be distinguished, model further proposes Latent Dirichlet Allocation based topic modeling technique to find hashtag of a tweet. The proposed method can suggest suitable hashtags in an fully unsupervised settings for 1.8 million tweet data where the hashtags were validated by consulting two experts.

(Guo, Liu, and Diab 2014) proposes to find similarity between two tweets that share same hashtag by calculating the hamming distance between unique binary code of tweets. This binary code is generated from latent representation of words in a tweet that have been learned by weighted matrix factorization of tf-idf matrix of tweets. An detailed comparison of all the above methods can be found in table 1.

Though above methods are efficient in finding related tweets, they have some drawbacks: (i) they consider one tweet during the learning phase and can only find similarity between tweets *after* learning phase is over. Ignoring the relational features of the data during learning can harm the performance of the learned model. The proposed model (RDN-Boost) considers the relation between tweets *during* learning. (ii) Most of the models discussed above have to learn and hence store millions of parameters of the model e.g embeddings vectors, weights of logistic regression etc, whereas proposed model is non-parametric model.(iii) Most of the models discussed above are uninterpretable as they can not explain why two tweets are related whereas the proposed model can produce human readable rules explaining why two tweets are related. In the next section, we explain the RDN-Boost model in detail that have been utilized to learn relations between tweets in social media data.

RDN-Boost Model

Dependency Networks(DN) (Heckerman et al. 2001) are probabilistic graphical models, which are an alternative to Bayesian Networks. However, unlike Bayesian Networks, Dependency Networks allow cyclicity in their structure.

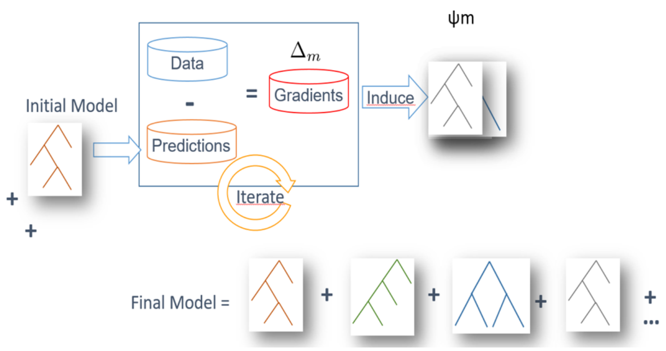

Figure 1: RDN-Boost. The functional gradient boosting in RDN-Boost is similar to standard gradient-boosting where trees are induced stage-wise manner.At every iteration, the gradient are computed as difference between observed and predicted probabilities of each example and a new regression tree is fitted to these examples

(Neville and Jensen 2007) extended Dependency Networks to relational case by proposing Relational Dependency Networks (RDN). RDN learns a set of conditional distribution where each distribution is represented using relational probability trees. Motivated by the intuition that finding many rough rules of thumb of how to change one's probabilistic prediction locally can be a lot easier than finding a single, highly accurate local model, (Natarajan et al. 2012) proposed RDN-Boost that turn the problem of learning RDN into a series of relational functional approximation problems using gradient based boosting. RDN-Boost model is a relational functional gradient boosting model (RFGB) and is explained below.

The goal of this project is to learn whether two tweets are related with each other from the text of the tweet. Following standard machine learning notations, we denote $\texttt{related}(\texttt{tweet1}, \texttt{tweet2})$ (or `related` as abbr.) as y and all the text features of tweet as $\mathbf{x}$. The key idea in gradient boosting is to consider functional representation for distribution and derive the gradients of log-likelihood wrt function (Natarajan et al. 2017). RDN-Boost represents a probability distribution as sigmoid over regression function ψ:

$$P(y = \texttt{related} \,|\, \mathbf{x}) = \frac{e^{\psi(y = \texttt{related}, \mathbf{x})}}{1 + e^{\psi(y = \texttt{related}, \mathbf{x})}} \quad (1)$$

Standard gradient descent methods take the gradient of log-likelihood wrt the parameters of distribution. The log-likelihood in standard methods is given by LL = $\sum_i log(P(y_i|\mathbf{x}))$. RDN-Boost, on the other hand, takes the gradient of this log-likelihood wrt ψ. (Friedman 2000) suggested computing gradient for each example separately and fit a regression function over all weighted examples. Hence, key insight is that gradients are not summed over all examples, but instead, are computed for each example and this gradient becomes the weight of example ($\Delta(x_i)$). This set of local gradients will approximate the global gradient. The functional gradient of each example <$\mathbf{x}_i$, y_i> wrt regression function ($\psi(y_i = 1|\mathbf{x}_i)$) is:

$$\frac{\partial \, log \, P(y_i|\mathbf{x}_i)}{\partial \psi(y_i = 1|\mathbf{x}_i)} = I(y_i = 1|\mathbf{x}_i) - P(y_i = 1|\mathbf{x}_i) \quad (2)$$

where I is the indicator function that is 1 if y_i = 1 and 0 otherwise. The expression is simply adjustment required to match predicted probability with the true label of the example. If example is positive (two tweets are related) and the predicted probability is less than 1, this gradient is positive indicating that the predicted probability should move towards 1. Conversely, if example is negative (tweets are not related) and the predicted probability is greater than 0, the gradient is negative, pushing the value other way.

Now, to fit the gradient function for every training example, we use Relational Regression Trees (Blockeel and Raedt 1998). Similar to standard gradient descent, the sum of m gradients is current value of regression function ψ. Note that when given an example and value of all features, only one path in the tree is satisfied (Yang et al. 2014). Then regression value from the leaf of that path is taken as gradient and added to the overall gradient. Hence these different trees can be summed to obtain ψ value of the current example given its features. This is an iterative procedure where at each step, the gradient computed for each example and a relational tree is learned over all the examples. Then this tree is added to the current set of trees and the procedure continues till convergence (or when preset number of trees are learned).

Interpretability of RDN-Boost Model

In addition to its state-of-the-art performance, RDN-Boost, or SRL models in general, have an advantage of interpretability over deep models. Whereas with deep models it is difficult to interpret what weights at the hidden layers of the neural networks signify, it is easier to interpret the clausal (or logical) nature of SRL models.

However, interpretation of multiple trees learnt in RDN-Boost is still an issue. Even though state-of-the-art results are obtained with RDN-Boost, this model still can not directly perform qualitative analysis of the results. This is because the newer trees learnt in current iteration of RDN-Boost depends upon the trees already learnt in the previous iteration i.e., like boosting they "fix" the errors due to the previous trees. Hence, all the trees learned in RDN-Boost are not independent of each other. More importantly, since they are all weak learners, we can not consider a few of relational trees in order to reason about the presence of relations existing in the data (Natarajan et al. 2017).

RDN-Boost's implementation takes an approximate approach, that they call as Craven approach (Craven and Shavlik 1995), to generate a combined relational tree from all the weak relational trees learnt in RDN-Boost. This approach was originally used for making interpretable neural networks. According to this approach, after learning an RDN-Boost model on training data, we use this learned model to relabel the training data and once again learn an overfitted tree to this newly labeled data. The results obtained by this newly learnt tree would be representative of original RDN-Boost's model because the relabeling of data was performed by utlizing the original model. Our original training data consists of Boolean labels (related vs unrelated tweets). But the relabeled data consists of regression values that are being learned in the new tree. Hence, the resulting tree is closer to the original learned model (Natarajan et al. 2017).

Data Description

Given that we have outlined the learning algorithm considered in the previous section, we now explain the data collection from twitter in detail. This is further divided into two subtasks: getting the data from twitter and bringing it to the format that is acceptable to RDN-Boost.

Data Crawling and Cleaning

We crawl data from twitter API[1] by making an assumption that related tweets share a hashtag while unrelated tweets do not. Further, we aim to find if tweets are related or not based solely on the text of the tweets. We selected two unrelated trends : $\#Sridevi$ and $\#MondayMotivation$ that were trending in India on 26th Feb, 2018. We crawled the tweet text from these hashtags. Sridevi was an Indian movie star who suddenly died after accidental drowning

[1]https://dev.twitter.com/docs/api/1.1/overview

Table 2: Results of executing RDN-Boost on twitter data for different experiments. It can be seen that RDN-Boost exhibit excellent performance across all experiments.

Experiment	AUC-ROC	AUC-PR	Precision	Recall	F1 score
EXPERIMENT 1	0.973	0.976	0.964	0.81	0.88
EXPERIMENT 2	0.971	0.973	0.943	0.83	0.883
EXPERIMENT 3	0.958	0.902	0.858	0.97	0.911

on 24th Feb, 2018. Tweets in $\#Sridevi$ express shock over her sudden death and offer condolences to her family. $\#MondayMotivation$, on the other hand, comprises of positive and motivational tweets.

After crawling tweet text from twitter, top 600 tweets that have maximum retweets were selected from each hashtag. Each tweet has a unique `tweetid` associated with it. In addition to text data, we also considered emojis, emoticons, hyperlinks and mentions in a tweet as the features of tweets. We further cleaned the tweet data in order to remove all the special characters from them. We replaced emojis, emoticons by their corresponding names. For this purpose, we use two packages: `emoji`, `emot`[2] that convert each emoji to its respective name by using `emoji.demojize()` function. We also replaced all the special characters in the hyperlinks by their corresponding names (e.g. `http://abc` was replaced by `http_colon_fslash_fslash_abc`). We also retained the mentions of twitter handles (e.g. `@realDonaldTrump`) in each tweet's text. After cleaning the data, we bring the data into form that is acceptable to RDN-Boost, which is explained in the next section.

RDN-Boost Acceptable Data

From the data obtained in the previous section, we randomly split 600 tweets in each hashtag as training and test tweets in the ratio 5:1 respectively. RDN-Boost requires three files for training as well as testing: (a) positive examples (b) negative examples. Both positive and negative examples are of the form `related(tweetid, tweetid)`. (c) facts that represent the features of the model. We can generate positive examples by generating all pair of tweets present in either $\#Sridevi$ or in $\#MondayMotivation$. We have generated positive examples in both ways and have done two different experiments by considering each of them as explained in the next section. Negative examples represent pair of unrelated tweets that do not share a hashtag. We have generated negative examples by pairing each `tweetid` in $\#Sridevi$ with each `tweetid` in $\#MondayMotivation$.

To generate facts (features), we split the tweets on white spaces and used each word as a feature of the data that would be fed to RDN-Boost. Each word is encoded as `hasword(tweetid, wordname)` in the model. We also generated features for emoticons, emojis, mentions and hyperlinks. The facts file of RDN-boost contains the following features written in first-order logic form: `emoji(tweetid, emojiname)`, `hashtag(tweetid, hashtagname)`, `emoticon(tweetid, emoticonname)`, `mentions(tweetid, mentionname)`, `hyperlink(tweetid, hyperlinkname)`, `has_word(tweetid, wordname)`, `has_emoji(tweetid)`, `has_emoticon(tweetid)`, `has_hyperlink(tweetid)`, `has_hashtag(tweetid)`, `has_mentions(tweetid)` where `has_emoji`, `has_emoticon`, `has_hashtag`, `has_hyperlink`, `has_mentions` are the features that informs about the presence or absence of at least one emoji etc in a tweet. The above features were extracted for both $\#MondayMotivation$ and $\#Sridevi$ hashtags and written to facts file. The test data is generated in the same way as train data.

After generating the data, `hashtag(tweetid, MondayMotivation)` and `hashtag(tweetid, Sridevi)` were discarded from the facts file because their sole purpose was to act as a gold standard for generation of data. We perform the experiments to evaluate the performance of RDN-Boost on the resulting twitter data as explained in detail in the next section.

[2] `https://pypi.python.org/pypi/emot/1.0`

Experiments

In our experiments, we aim to ask the following questions:

Q1: Is RDN-Boost a viable tool for predicting the relation between tweets on Twitter ?
Q2: Does the presence of hashtags as the features aid in improving the performance of model ?
Q3: Does the method of generating positive examples affect the performance of the model ?

We perform an experiment to answer each of the above question. To answer **Q1**, we train RDN-Boost model on the data where positive examples are generated by pairing tweets belonging to $\#Sridevi$. We call it **Experiment 1**. Further, we examine if the approach of positive example generation affects the results (**Q3**) produced by model. To evaluate that, we perform another experiment (**Experiment 3**) by generating positive examples by pairing the tweets in $\#MondayMotivation$ and keeping negative examples and features same as in experiment 1.

Further, even though we remove `hashtag(tweetid, Sridevi)` and `hashtag(tweetid, MondayMotivation)` from data, there might still remain some hashtags that are semantically close to above two hashtags. The presence of these `hashtag(tweetid, hashtagname)` features might trivialize the learning phase of RDN-boost model. For instance, `hashtag(tweetid, RIPSridevi)` or `hashtag(tweetid, MondayMotivator)` are still present in the data and if any two tweets share `hashtag(tweetid, RIPSridevi)` feature then it is easy for the model to learn that they are related tweets. Hence, to verify how presence of such hashtags affect the performance of model (**Q2**), we perform another experiment (**Experiment**

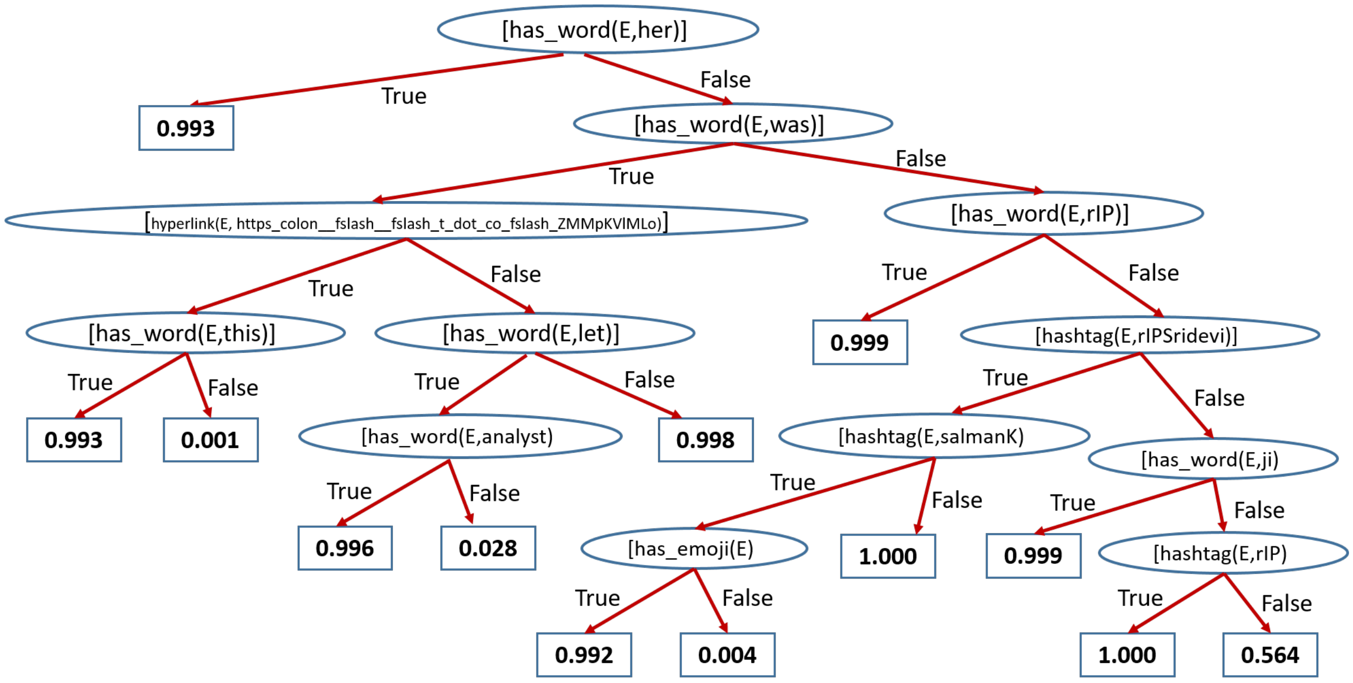

Figure 2: The combined tree learned by RDN-Boost for finding related tweets on Twitter. The leaf of the tree represents the probability of that branch of tree being true

2) by disabling all the hashtag(tweetid, hashtagname) features in facts file generated in the previous section. The final statistics of train data is shown in table 3. The test data has 10K positive examples, 10K negative examples and 4K facts for all the three experiments.

Table 3: The train data used to learn RDN-Boost model. $\#MonMot$ stands for MondayMotivation. Exp. is abbreviation for Experiment. Pos Generation is the Hashtag whose tweets were paired to generate positive examples.

Experiment	$\#Facts$	$\#Pos$	$\#Neg$	Pos Generation
Exp. 1	20.8K	0.25M	0.258M	#Sridevi
Exp. 2	20.1K	0.25M	0.258M	#Sridevi
Exp. 3	20.8K	0.266M	0.258M	#MonMot

Parameter Setting

Implementation of RDN-Boost allows us to set many flags of the model like depth of regression trees, bridgers, number of features inside a single node of the tree, number of clauses learnt by tree etc. that can be set to improve the performance of the model. We, on the other hand, adhere to the default settings of the model in all the experiments conducted in this paper. Our model learns 20 relational regression trees to learn gradient function of RDN-Boost.

Evaluation Metrics

We have considered different standard metrics to quantitatively analyze the results. In addition to standard Area under ROC curve, we also considered AUC-PR curve that is a important metric when we have skewed data in relational domains. Even though the test data is perfectly balanced, we report AUC-PR because it is generally reported for relational model. We also report precision and recall as our metrics. Finally we also report the F1 score on the results.

Results

Implementation of RDN-Boost is a publicly available software[3]. Even though python version of RDN-Boost is recently made available[4], for this project we relied on the java version of the project as it has been rigorously tested over many years. After learning RDN-Boost model on twitter data, the results of the test performance of the model is shown in table 2. As can be seen from the Experiment 1 results, RDN-Boost performs outstandingly well across all the metrics. Hence we can conclude that RDN-Boost is successfully able to learn relations present between tweets in social media data. Therefore, **Q1** can be answered affirmatively.

Further, as can be seen in results of experiment 2, the absence of hashtag(tweetid, hashtagname) features from the data does not affect the performance of the model compared to the results of experiment 1. When the RDN-Boost model does not receive hashtag features, it replaces all the features in relational trees by has_word features. This allows model to perform well without any deterioration in performance. Hence **Q2** can be answered negatively as absence of hashtag features does not affect the model as long as good textual features are present in the model that can help RDN-boost model to find relation between two tweets.

To answer **Q3**, we can see from experiment 3 results that when the positive examples of model are selected as tweet

[3]https://starling.utdallas.edu/software/boostsrl/
[4]https://starling.utdallas.edu/software/rfgb/

pairs in $\#MondayMotivation$ rather than $\#Sridevi$ as in experiment 1, the precision of the model drops considerably. It can be inferred from these results that there is not prominent features shared among the tweets of $\#MondayMotivation$ that can distinguish them from the tweets of $\#Sridevi$. Hence RDN-Boost is misclassifying two unrelated tweets as related (false positives) and hence that results in drop of the precision value. We could conclude that the type of data - whether tweets in a hashtag strongly share common information that distinguishes it from tweets in other hashtag - affects the performance of the model.

We also present the tree learned after combining the boosted trees from RDN-boost in figure 2. This tree is learnt from the data in experiment 1 where pair of tweets in positive example share $\#Sridevi$. The positive or negative example is $\mathtt{related(D,E)}$ where variable $\mathtt{D}$ and $\mathtt{E}$ represent tweetid of two tweets. In case of negative examples, first argument (i.e. $\mathtt{D}$) refers to $\#Sridevi$, whereas second argument (i.e. $\mathtt{E}$) refer to $\#MondayMotivation$. As a general rule, the left branch indicates that the test is satisfied and the right branch is when it is not. Also the leaf of each branch informs about the probability of that branch being true.

Some interesting characteristics of the tree is that most frequent and hence most important feature present in the tree is $\mathtt{has\_word(tweetid, word)}$. This suggests that the textual content of pair of tweets is most significant characteristic in finding the relation between tweets. If two tweets have same textual content, they are discussing same topic and hence are related with each other.

Second most common feature in the tree is $\mathtt{hashtag(tweetid, hashtagname)}$, followed by $\mathtt{hyperlink(tweetid, hyperlinkname)}$ and $\mathtt{has\_emoji}$ $\mathtt{(tweetid)}$. $\mathtt{emoticon(tweetid, emoticonname)}$ and $\mathtt{emoji(tweetid, emojiname)}$ have not been picked by the RDN-boost model among the most important features. The link present in `hyperlink` feature in the tree is the link to the video of the last public appearance of Sridevi before she died. Twitter users shared this link in their tweets when paying tribute to her.

The way to understand this tree is that if second argument of an example $\mathtt{related(D,E)}$ (i.e $\mathtt{E}$ here) has word 'her' in it, which in this case is referring to Sridevi, then the two tweets are related with probability 0.993. Another branch in the tree indicates that if a tweet in second argument does not contain word 'her', but contains word 'was' and contains the hyperlink to her last public appearance and also contains word 'this' (which perhaps is referring to the hyperlink), then probability of the tweets being related is 0.993. This makes sense because positive and negative examples are distinguishable from each other by second argument of $\mathtt{related(D,E)}$ ($\mathtt{E}$ refers to $\#MondayMotivation$ tweets for negative examples and $\#Sridevi$ for positive examples). For unrelated tweets, $\mathtt{E}$ will not refer to any 'her' mostly and this has been picked by the model to distinguish between positive and negative examples.

Without discussing the rest of the tree, we can see that the tree has interesting features that can allow us to find if two tweets are related or not. Slightly different trees are learned for experiment 2 and experiment 3 that can be seen in the code submitted with this paper. In addition, the model also outputs the probabilistic score of each test example at test time that informs about the amount of relatedness between two tweets. These probabilistic scores for each test example can be seen in the results submitted with this work.

From the above experiments, it can be concluded that RDN-Boost is a powerful model that can extract relational information from social media data. In addition to its performance, we also obtain interpretable boosted trees that can reason about why two tweets are related with each other and also by what probability. This knowledge of finding related tweets on Twitter can be utilized for many purposes: if two tweets are related, tweeters of the tweet have similar opinion about some topic. Hence, this can be used as recommender for two users to follow each other or recommend one use to follow the people that second user is following.

In spite of performing so well on the social media data, the above experiments have a drawback that they have not been validated by k-fold validation and hence the results are not as generalizable. Though we can perform k-fold cross validation, by randomly splitting 600 tweets into train and test k times and perform our experiment k time on each of resulting train and test split, we leave it as future work.

Second disadvantage of this model (and SRL model in general) is that this model takes considerable amount of time for training. For instance, the learning phase of all the experiments required two days for completion. If we further increase the dataset size by further considering, say 10K tweets (rather than 600 tweets considered in this work) to find the relations between tweets, RDN-Boost model might take weeks for training on large data. Thirdly, the combined tree learnt in figure 2 is an approximation of the exact trees learnt by RDN-Boost which might result in some loss of information during approximation.

Conclusion

We considered the problem of finding related data on twitter based upon the features of words, emoticons, hyperlinks etc. present in the tweet of text. We applied an advanced SRL model named RDN-Boost on this data that produced excellent results. RDN-Boost model is interpretable, can handle uncertainty and considers relational aspect of data *during* training rather than *after* training as in models discussed in the related work. To the best of our knowledge, this is the first application of any SRL model on twitter data.

An important limitation of our current approach is that we treat the hashtags as ground truth and they are essentially a proxy for the true labels. Modeling the accuracy of this approximation remains an interesting future challenge. As a future avenue, the advanced flags of RDN-Boost can be set to see how that improves the performance of the model. Also, k-fold cross validation needs to be performed on the data to generate more general results. Further, the model can be applied to large number of tweets. Most importantly, successful application of SRL model on social media data in this project opens the door to applicability of SRL models on other applications, like to perform sentiment analysis, emotion detection from social media data.

Team Member Contribution

I worked alone on this project.

References

Blockeel, H., and Raedt, L. D. 1998. Top-down induction of first-order logical decision trees. *Artificial Intelligence* 101(1–2):285–297.

Craven, M. W., and Shavlik, J. W. 1995. Extracting tree-structured representations of trained networks. In *NIPS*, 24–30.

Dhingra, B.; Zhou, Z.; Fitzpatrick, D.; Muehl, M.; and Cohen, W. W. 2016. Tweet2vec: Character-based distributed representations for social media. In *Association for Computational Linguistics*, 269–274.

Friedman, J. H. 2000. Greedy function approximation: A gradient boosting machine. *Annals of Statistics* 29:1189–1232.

Getoor, L., and Taskar, B. 2007. *Introduction to Statistical Relational Learning*. The MIT Press.

Godin, F.; Slavkovikj, V.; De Neve, W.; Schrauwen, B.; and Van de Walle, R. 2013. Using topic models for twitter hashtag recommendation. In *International Conference on World Wide Web*, 593–596.

Goel, A.; Sharma, A.; Wang, D.; and Yin, Z. 2015. Discovering similar users on twitter. In *Workshop on Mining and Learning with Graphs (MLG-2013)*.

Guo, W.; Liu, W.; and Diab, M. T. 2014. Fast tweet retrieval with compact binary codes. In *International Conference on Computational Linguistics*, 486–496.

Heckerman, D.; Chickering, D. M.; Meek, C.; Rounthwaite, R.; and Kadie, C. 2001. Dependency networks for inference, collaborative filtering, and data visualization. *Journal of Machine Learning Research* 1:49âĂŞ–75.

Jensen, D., and Neville, J. 2002. Linkage and autocorrelation cause feature selection bias in relational learning. In *ICML*, 259–266.

Le, Q., and Mikolov, T. 2014. Distributed representations of sentences and documents. In *International Conference on Machine Learning*, 1188–1196.

Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G. S.; and Dean, J. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26*, 3111–3119.

Natarajan, S.; Khot, T.; Kersting, K.; Gutmann, B.; and Shavlik, J. 2012. Gradient-based boosting for statistical relational learning: The relational dependency network case. *Machine Learning* 86(1):25–56.

Natarajan, S.; Prabhakar, A.; Ramanan, N.; Baglione, A.; Connelly, K.; and Siek, K. 2017. Boosting for postpartum depression prediction. In *CHASE*, 232–240.

Neville, J., and Jensen, D. 2007. Relational dependency networks. *Journal of Machine Learning Research* 8:653–692.

Severyn, A., and Moschitti, A. 2015. Twitter sentiment analysis with deep convolutional neural networks. In *ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '15, 959–962.

Weston, J.; Chopra, S.; and Adams, K. 2014. #tagspace: Semantic embeddings from hashtags. In *EMNLP*, 1822–1827.

Xu, W.; Callison-Burch, C.; and Dolan, B. 2015. Semeval-2015 task 1: Paraphrase and semantic similarity in twitter (PIT). In *International Workshop on Semantic Evaluation*, 1–11.

Yang, S.; Khot, T.; Kersting, K.; Kunapuli, G.; Hauser, K.; and Natarajan, S. 2014. Learning from imbalanced data in relational domains: A soft margin approach. In *International Conference on Data Mining (ICDM)*, 1085–1090.

Zarrella, G.; Henderson, J.; Merkhofer, E. M.; and Strickhart, L. 2015. Mitre: Seven systems for semantic similarity in tweets. In *International Workshop on Semantic Evaluation (SemEval 2015)*, 12–17.