An Novel RNN Approach to Classification of Complex Textual Scientific Metadata

Reid McIlroy-Young MACS 302 Final Poster June 1, 2017

Introduction

Currently most analysis of scientific publications is limited to those fields available in the database used by the researchers. Efforts to extend the fields usually rely on unsupervised clustering (e.g. Boyack, *et al*, 2005). Presented here is a new technique for using supervised classification on scientific metadata, with imperfect training data. By using a complex model that is reading the unstructured text we can define much more abstract concepts than a similarity measure. For this analysis we looked for *papers introducing new software packages, tools or interfaces.*

Data

Our data are all articles published in the top 123 statistics journals, as identified by the Web of Science (WOS), between 2005 and 2015, giving a total of 78 971 articles. The data were provided by the WOS mirror database hosted by Knowledge Lab. From these, three journals were identified that primarily introduce new software and eight that almost never do so. These gave us 1251 positive examples and 4362 negative for training. Because we are using journal names alone and no hand coding some small fraction of our training data is incorrectly labelled.

Computation Graph

The neural network (NN) can be understood as a computation graph with the weights being computed via backpropagation through the graph. Figure 1 shows the partiality unfolded graph, with the circular nodes being those learned in training.

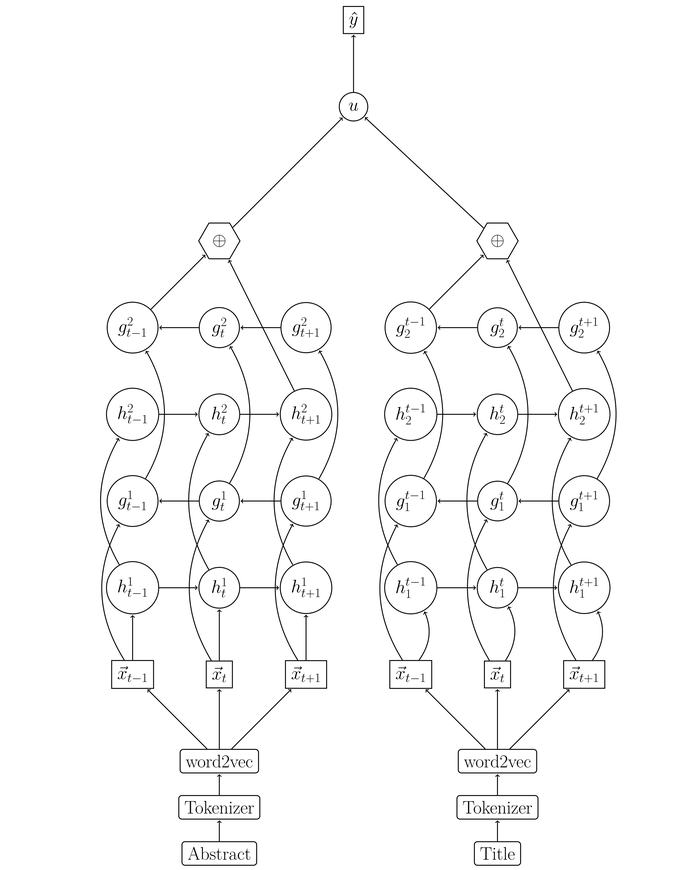

Figure: Simplified recursive bidirectional Recursive NN (RNN) layout, LSTM connections were removed for clarity. Circles are NN layers, rectangles with curved corners are the preprocessing, hexagons are combined outputs from the RNN layers, and final output and inputs are indicated with squared corners.

Methods

For our analysis we have two pieces of text for each article that we provide to the model, the abstract and title; since from these most humans can quickly identify if the article describes a new piece of software. Note these fields are both unstructured text with some even being non-English. To convert these into usable form we first tokenize them at the word and sentence levels, then trained a 200-dimensional Word2Vec embedding model from all tokens in the data set. Note that, no tokens are dropped at any point, the Word2Vec model is trained with both punctuation and highly infrequent tokens.

Our classifier is a Recursive NN composed of two parallel bidirectional two-layer recurrent NNs (RNNs) using Long Short-Term Memory (LSTM) cells whose outputs are provided to a final single fully connected layer. Both input layers have 200 nodes, the hidden RNN layers are all 256 nodes and the final output is 2 nodes. The inputs to each of our RNNs are sequences of Word2Vec vectors for each of the word level tokens, sentence levels are ignored but all punctuation has a Word2Vec vector so the RNN can learn to recognize sentences.

To train the model the weights were updated with automatic backpropagation against the cross entropy loss. Since the positives are greatly outnumber by the negatives we added them twice to the training set, so there was a 36.8% chance of a positive example being chosen for a batch. 10% of the testing set was held out for cross-validation and the final results are shown in Table 1.

Epochs	8
Batches per Epoch	500
Testing Loss	0.093
Testing Error Rate	0.023
Detection Rate	0.955
False Positive Rate	0.017

Table: Testing results for optimal model

RNN Layers

Getting a model that can make classifications with a 4% error rate is very good, but we want to know if the model has some understanding of its inputs or if it is simply looking for words (e.g. '*R*') or sequences of words (e.g. '*a new package*'). To do this we can look at the outputs from the RNN layers to the fully connected layer u, these are updated for each word and only the final values are provided to the hidden layer, specificity we are seeing the outputs from g^2 and h^2 in figure 1 concatenated into a 512 dimensional vector. Figures 2 and 3 show the activations for each word in the title and abstract respectively, of a positive and negative sample. We can see by examining these plots that both RNNs have somewhat nuanced understandings of their inputs.

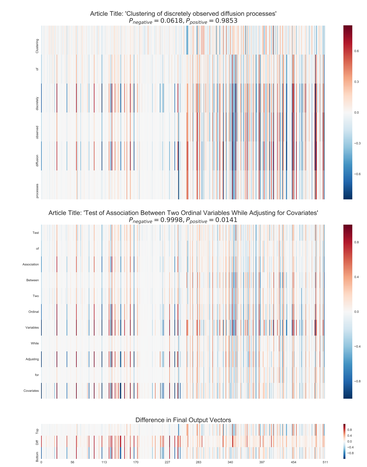

Figure: RNN activations for each word in two titles; the positive example is on top, the negative below it, and a comparison of each input's final output is shown at the bottom.

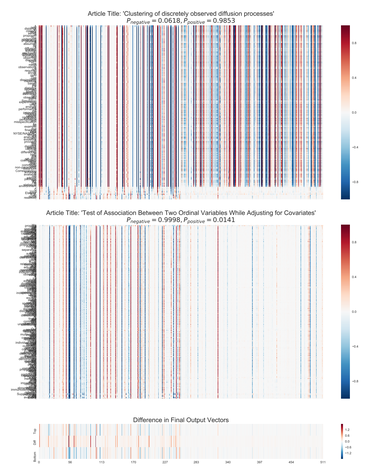

Figure: RNN activations for each word in two abstracts; the positive example is on top, the negative below it, and a comparison of each input's final output is shown at the bottom.

Year	Number of Articles	Not Software	Software	New Software
2005	5 235	4 998	237	4.52%
2006	5 806	5 522	284	4.89%
2007	6 255	5 914	341	5.45%
2008	7 085	6 703	382	5.39%
2009	7 435	7 047	388	5.21%
2010	7 216	6 801	415	5.75%
2011	7 767	7 280	487	6.27%
2012	7 990	7 519	471	5.89%
2013	8 287	7 829	458	5.52%
2014	8 336	7 830	506	6.07%
2015	7 559	7 095	464	6.13%
Total	78 971	74 538	4433	5.61%

Table: Per year result across all papers

Highest software count (Training)	673	JOURNAL OF STATISTICAL SOFTWARE
Highest software count (Non-training)	171	IEEE-ACM TRANSACTIONS ON COMPUTATIONAL BIOLOGY AND BIOINFORMATICS
Highest non-software count (Training)	1 156	ANNALS OF STATISTICS
Highest non-software count (Non-training)	3 381	STATISTICS & PROBABILITY LETTERS

Table: Top journals for each class in training and full dataset

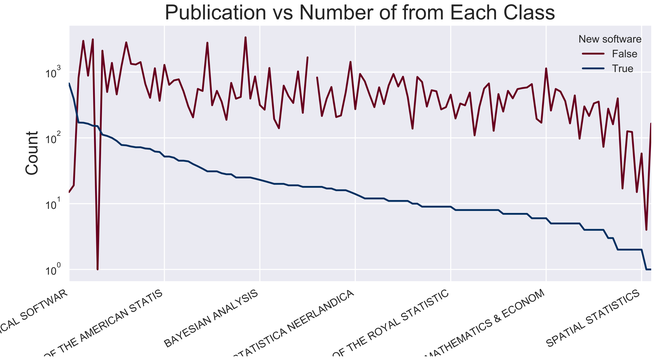

Figure: Counts for each class against each journal, ordered by number of positive examples, note y axis is log

Discussion

As we have shown, the model is capable of identifying complex ideas in textual data. The training set is both relatively small and imperfect; in fact, when the false classifications are examined by an expert, most of them result from errors in the data and not in the model. Thus, by using the model to help generate cleaner data, a new model could be trained with higher accuracy. The main issue is the false detection rate, which could be reduced by training a cascade of NNs instead of a single one.

These issues aside, the research undertaken herein demonstrates large-scale detection of new software introduced in scientific literature. The next steps are to do biometric and natural language processing on our corpora as there are many questions we can answer. Of interest to us are '*How programming languages distributed across the sciences?*', '*What properties of new scientific software packages cause them to be accepted by a community?*' and '*What is the relationship between success as an article and success as a software tool?*'.

Source

The code used for this analysis is available at: github.com/reidmcy/New-Software-Identification