Luke Sturgeon

Stat 517

11/15/18

Literature Review: Topic detection using paragraph vectors to support active learning in systematic reviews

**Intro**

To take advantage of the all the published literature written on topics, easily available to researchers now through online journals and libraries, systematic reviews are performed to glean information from a wide range of papers. While any one paper may cite tens of documents itself, systematic reviews are much larger in scope, typically encompassing thousands of articles. Because of this, however, many more resources must be devoted to the review; specifically, time. While one researcher may be tasked with going through each abstract by hand and deciding whether it should be included in the review, often a second or third researcher does the same so that an interrater reliability can be found. This ensures the right articles are being included in the final review. This prescreening process of the abstracts thus takes hundreds of hours to complete and the researchers must still do a final screening of articles before beginning the data analysis from each paper. To hasten this process, prescreening tools have been developed using machine learning to group clusters of articles, effectively prescreening the corpus for the researchers. One such prescreener is RobotAnalyst (RA). RA is unique, using paragraph vectors to model abstract/title pairs (referred to as documents hereafter) to try and capture contextual clues other methods do not. This has been shown to be especially effective in clinical and public health articles (Hashimoto et al., 2016).

**Overview**

This paper will follow the steps RA takes in prescreening articles from the user’s perspective. At each step of the process, the models that are being developed and trained by RA (inside the black box, for lack of a better phrase) will be explained. Afterwards, data will be given to compare RA performance to other topic detection methods.

**Process**

*Step 1*

The first step the user takes in this process is to gather abstracts and titles of the articles that could possibly be included in the review. 60 (or more, the amount only improves the training set of RA, explained later) of these documents are classified by the user; either should be included in the final review or shouldn’t. The documents that have been classified as included, the documents classified as excluded, and the documents that are unclassified all go into separate .ris files, a filetype used by librarians and bibliographic programs. These files are uploaded to the RA server via web browser. No executable file must be downloaded and installed on the user’s computer. All the computation is performed by servers curated by RA’s creators, the National Center for Text Mining (NaCTeM) in Manchester, UK.

*Step 2*

Once the files are uploaded, RA arranges the unclassified documents by level of confidence. It does not arrange them based on inclusion/exclusion, but rather how confident the model is that the article should belong to that classification. How it does this, briefly, is through a supervised learning process. RA has taken the pre-classified documents and trained ultimately logistic regression and Support Vector Machine (SVM) models to label each abstract/title pair for inclusion or exclusion.

How it does this takes several steps. The first step within the black box, what the user doesn’t see RA doing, is to build word vectors. To do this, the unclassified documents go through GENIA tagger. This proprietary program marks each word in each title and abstract with the part of speech and lemma of the word. This attempts to capture some of the semantic meaning of the word, not only the frequency with which it is found in the corpus. These same documents are then put through TerMine, another proprietary program that find multi-word phrases that are unique to each abstract and title. For example, when analyzing the abstracts regarding fatigue in miners, the phrase “structural fatigue” would be helpful in training models to exclude these abstracts from the final review since this article discusses the wrong type of fatigue.

Next, the conditional probability that these words will appear given the words preceding it in the article are calculated. Specifically,

*p*(*wt*|*wt*-*N*,*wt*-*N*+1,…,*wt*-1)

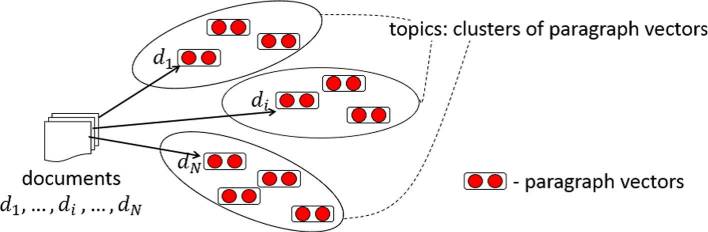
where *wt* is the target word and *wt*-*N*,*wt*-*N*+1,…,*wt*-1 are *N* context words that occur before *wt* (Hashimoto et al., 2016). By including the relevancy of the preceding words in the probability of the target word, these word vectors are attempting to include more semantic meaning. These probabilities are continuously updated, each abstract/title pair updating the parameters of the word vector. Thus, word vectors are shared between all documents. To clarify, target words are going to be most probable when a certain list of other target words precede it in the document. If “pet,” “veterinarian,” and “unconditional love” precede the target word, it would be probable that “dog” would be the most likely word vector to follow those words. Other target words that are probable in that same location are close in vector space since their word vectors behave similarly to cat. So the word vector for “dog” would find itself neighbors with “cat” since it would be reasonable that “pet” and “veterinarian” may be associated with “cat” as well. These continuously improved word vectors have been shown to be superior to the typical bag-of-words methods that put every word at equal length from each other in vector space (Le, Mikolov, 2014). Although not specified in this article, the method in which these word vectors are trained has been done with multiclass classifiers such as soft max, as done in a paper by Le and Mikolov, 2014, which is cited by this paper when discussing the creation of word vectors. Code that is typically used for this training can be found at <https://code.google.com/archive/p/word2vec/> (Le, Mikolove, 2014).

What separates RA from other prescreening software is that it uses both paragraph and word vectors jointly in the same neural network when grouping documents. Paragraph vectors are trained in a similar way to word vectors, only the document the word vectors are in are considered now. Specifically,

p(wt|wt-N,wt-N+1,…,wt-1,d) = σ(s(wt) · [v(wt-N);v(wt-N+1);…;v(wt-1);v(d)])

where s(wt) is a weight vector for computing the conditional probability, σ( · ) is the logistic function, [ · ; · ] is the concatenation of vectors, and d denotes the document including the sequence of words (Hashimoto et al., 2016). These vectors are maximized, so that the target word is being weighted by the words closely related to it as well as the documents it’s typically found in. By relating word vectors and paragraph vectors, documents can now be put into vector space by their paragraph vectors closer to similar documents and further from dissimilar pairs. It is important to note that the distance used when building these vector spaces for word and paragraphs vectors is the cosine distance, rather than Euclidian or other forms (Le, Mikolove, 2014; Hashimoto, 2016).

Since these paragraph vectors have been built in such a way that semantic and contextual information isn’t lost, detecting latent topics within a document is much more accurate than bag-of-word approaches (Hashimoto et al., 2016). RA calculates the cosine distance between a paragraph vector and k number of centroids using spectral clustering. Those paragraph vectors found closer together are assumed to represent a topic.



(Hashimoto et al., 2016)

The preceding illustration gives a good visual aid in how topics are found by clustering documents which are, in turn, made of paragraph vectors. Once these clusters have been maximized, key words are found to describe the topic. Words that are most common to the topic, found by ranking the words with the highest conditional probability within the cluster, are shown as a list to the user. So a list of “drunk,” “alcohol,” and “car accident” will be found in the same topic, given the abstracts are discussing the causes of road deaths, e.g. These topics are associated with either exclude or include, the idea being that documents that find themselves in a cluster that is included in the final review will be marked include by RA. By default, RA assumes that the corpus has 300 topics (centroids) and replicates the spectral clustering 10 times, 100 iterations of expectation/maximization each.

This process would take exceedingly long if all keywords were vectorized. Instead, RA uses Conditional Mutual Information Maximization criterion (CMIM) to reduce the number of keywords needing to be vectorized. This criterion takes a word and finds the amount of variation explained by this word given the amount of variability already explained by the other words in the document (Fleuret, 2004). This conditional variation explanation reduces redundancy since a word may be singularly powerful, explaining .4 of the overall variability by itself but much less when used in conjunction with other powerful words. Since it’s not adding much to the model, it is removed, speeding up the process.

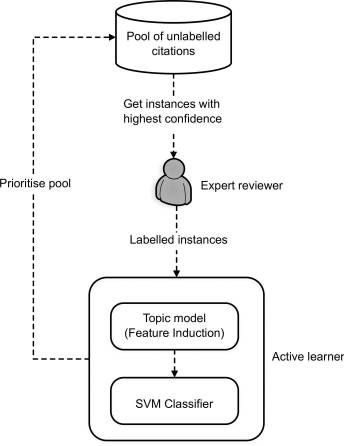
Now that topics have been clustered, RA trains an SVM and linear logistic regression model to classify the unclassified documents through LIBLINEAR, another program employed by RA. This increases the speed of processing significantly since these models are relatively simple in comparison to other machine learning models.

*Step 3*

These unclassified documents are listed in order of confidence. The user is encouraged to go through this ordered list and classify each abstract as included or excluded. This is what RA calls active learning. This process stops when the updated list (Step 4) has reached a 95% recall accuracy, better explained in the “Measurements” section of this paper.

*Step 4*

After the user has classified several articles, they can update the training set given to RA and the model is improved as the topics become more well defined. A larger pool of classified documents will naturally improve the model by training word and paragraph vectors to better represent topics to be included or excluded. This causes a human/machine learning loop that improves the process until the rest of the corpus can be classified with 95% confidence. The process is well explained by the following picture.



(Hashimoto et al., 2016)

**Measurements**

To make comparisons to other methods of topic selection and inclusion/exclusion accuracy, this article introduces a few measurements. The first is yield defined as

where N is the total number of citations, TP,TN,FP and FN the number of true positive (eligible studies), true negative (ineligible studies), false positive (studies that are incorrectly classified as eligible) and false negative instances (studies that are incorrectly classified as ineligible), where the superscript M and A denote manual and automatic screening decisions, respectively (Hashimoto et al., 2016).

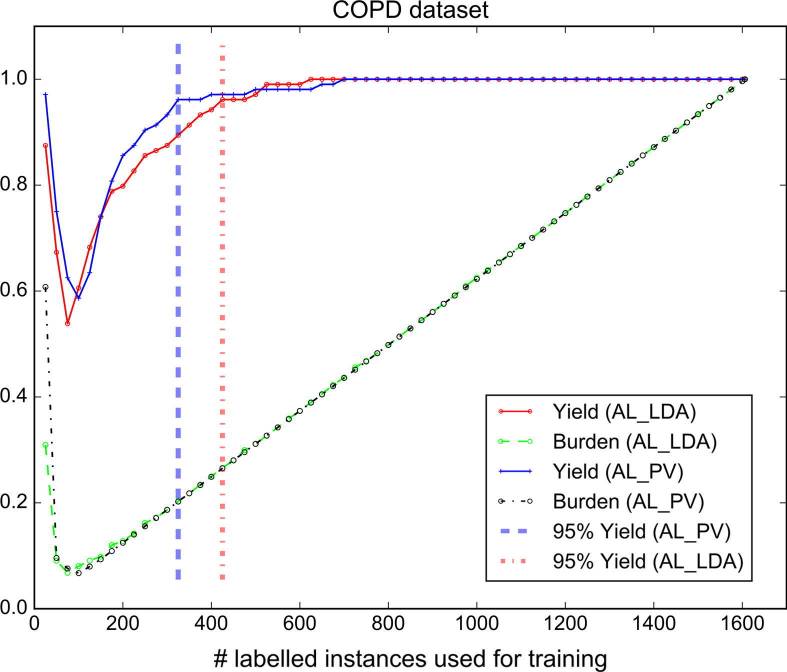
Second they introduce burden, defined by

where the variables are defined the same as for yield (Hashimoto et al., 2016).

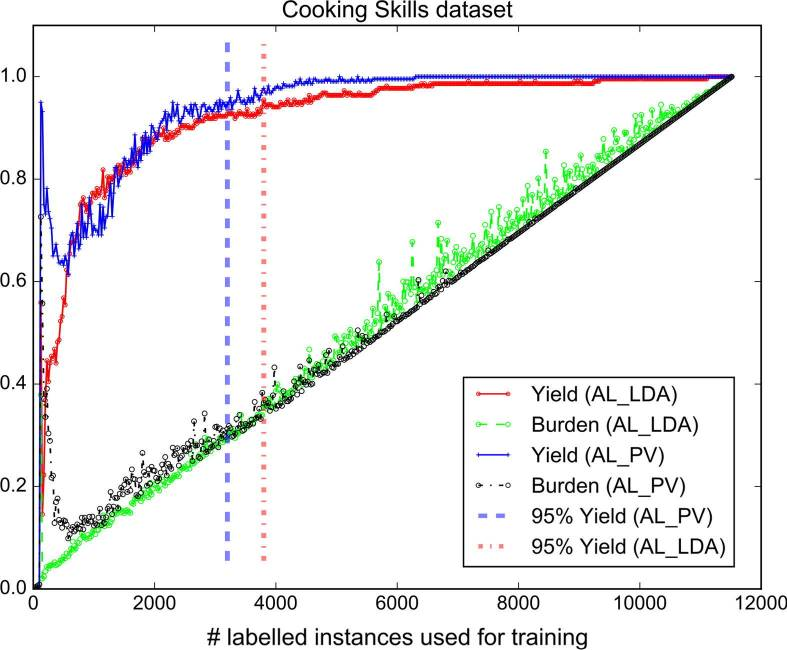
Lastly, work saved over sampling at 95% recall, WSS@95%, defined as

**Results**

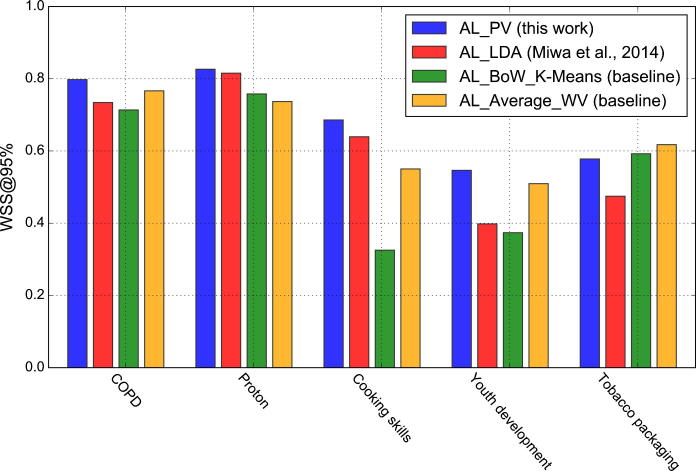
Several corpi were used for testing. Universities and government agencies performing systematic reviews were contacted and asked to help with proving the efficacy of RA. These universities and agencies went through each abstract manually, then RA was used to see what results could be achieved after the fact, quantifying the amount of time machine learning could have saved. Several methods of topic detection are used as comparisons to show the improvement RA affords researchers over these other methods.



(Hashimoto et al., 2016)

 The graph above describes a COPD corpus of literature. The y axis is the WSS@95% and the x is the number of classifications that had to be made manually. The green line shows the manual classification; for every article classified one article is classified correctly so we get our linear model as shown. RA, shown in blue shows that after roughly 260 classifications out of 1600, 95% accurate recall has been achieved. LDA, Latent Dirichlet Analysis, the other method of topic detection is show in red. It needed slightly over 400 manual classifications to provide as accurate a recall.

(Hashimoto et al., 2016)

Similarly, for a cooking skills corpus, RA out performs LDA. It achieves 95% recall with roughly 3100 manual classifications where LDA needs roughly 3900.

(Hashimoto et al., 2016)

For several corpi, several methods of topic detection were used to compare RA against. They measure the amount of work saved (percent classification made by the topic detection model with 95% accurate recall) for each dataset by each detection model. In all but one dataset, RA is more efficient. Over several datasets, RA was reported to increase performance with a significant difference, ranging from 5-15%, depending on the dataset (Hashimoto et al., 2016).

**Summary**

RA improves topic detection models by a modest 5-15%. This may sound like a minor increase in efficacy, but as seen in the “Cooking Skills” corpus, that improvement saved a human from having to classify roughly 800 abstracts. If each abstract takes 30 seconds to read and classify accurately, an assumption worth questioning after the researcher has gone through 1,000 abstracts, 6 hours 40 minutes could have been saved. Thus, RA allows researchers the ability to more efficiently use the vast literature that is out there regarding a subject, improving systematic reviews will minimal cost.

References

Fleuret, François. "Fast binary feature selection with conditional mutual information." *Journal of Machine Learning Research* 5.Nov (2004): 1531-1555.

Hashimoto, Kazuma, et al. "Topic detection using paragraph vectors to support active learning in systematic reviews." *Journal of biomedical informatics* 62 (2016): 59-65.

Le, Q., & Mikolov, T. (2014, January). Distributed representations of sentences and documents. In *International Conference on Machine Learning* (pp. 1188-1196).

Przybyła, P., Brockmeier, A. J., Kontonatsios, G., Le Pogam, M. -A., McNaught, J., von Elm, E., Nolan, K. and Ananiadou, S. (2018). Prioritising references for systematic reviews with RobotAnalyst: a user study.