Paper SAS3295-2019

The Wondrous New tmCooccur SAS® Cloud Analytic Services (CAS) Action and Some of Its Many Uses

James A. Cox and Russell Albright, SAS Institute Inc., Cary, NC

ABSTRACT

The new tmCooccur action in SAS® Viya® 3.4 detects significant co-occurrences of terms in sentences. It can identify patterns and nuances such as the following: 1) the same word used with different meanings (for example, "They met along the bank of a river" versus "They made a deposit in a bank"); 2) verbs used as regular verbs versus verb-particle combinations (for example, "shut the heck up" versus "shut the door please"); and 3) less common uses of negation, such as statements like "He liked the show very much" versus "He liked the show not at all". This paper describes how the results of the tmCooccur action can be used to generate word embeddings on the basis of local information (in a manner similar to GloVe and Word2Vec), and also how it can be used to create virtual terms that represent the most significant term co-occurrences. These new composite terms can then be used to create better topic, concept, and category definitions. Doing so can generate significant gains in lift for predictive modeling on some real-world data. Finally, this paper describes how to use the tmCooccur action with other types of transactional data, including IOT (Internet of Things) and genomic data.

INTRODUCTION

The term "word embedding" is the current buzzword in text analytics. A *word embedding* is a mapping of each word into a multi-dimensional space. Good word embeddings place similar words near one another in that space.

All word embeddings are based on the Distributional Hypothesis popularized by Firth (Firth 1957), namely that "a word is characterized by the company it keeps". Essentially, the embeddings for two words would be closest to each other when all other words in context with those words are the same. Traditionally, in SAS® Text Miner, this context has been taken to be the entire document in which a word appears.

In this paper, a new approach to word embeddings is presented that serves as a compromise between the strengths of the traditional SAS approach with the SVD and newer approaches that use shorter contexts than the entire document.

SAS TEXT MINER EMBEDDINGS WITH THE SVD

Word embeddings have been an integral part of SAS Text Miner from its inception through the Singular Value Decomposition (SVD) of the term-by-document matrix -- each cell in that matrix represents a weighted count of the number of times each word occurs in each document. This matrix is sparse and not very useful for any type of modeling on its own. Based on a user-specified "dimensionality" D, the SVD factors this original matrix into a matrix that contains D columns for each of the M terms that contain its points (or projection) into the D-dimensional space, and a second matrix that contains the projection of each document into that same space.

There is much that can be done once you have this joint "word embedding" and "document embedding". You can rotate the dimensions of the space to align with strong directions and call the resulting axes "topics" -- which provides interpretability to each of the dimensions. Or you can take the document embeddings, after suitably normalizing them, and feed them into any clustering or predictive modeling algorithm as predictor variables. Using these techniques within the flexible SAS® Enterprise Miner™ interface has been incredibly useful for our users over the years.

RECENT WORD EMBEDDING ALGORITHMS

More recently, techniques such as Word2Vec (Mikolov et al. 2013) and GloVe (Pennington et al. 2014) have been developed, which rely on a shorter context than an entire document, generally a sliding window of words, such that the context of a given word is only the three or five words immediately preceding it and the same number of words after it. The resulting embeddings, when built on very large corpora (such as Wikipedia), show significantly better results on semantic and syntactic word analogy tasks and word similarity tasks than embeddings like the term-by-document SVD decomposition with the entire document as context.

All the embedding techniques discussed so far create a single embedding for a given word (that is, surface text possibly combined with the part of speech tag). However, words can have multiple meanings (polysemy) depending on context -- for example, the noun "bank" can be either a lending institution or the point at which a river meets the land. For semantic models, this is problematic. Thus, in the last year, models that provide different embeddings for a given word depending on context, such as BERT (Devlin et al. 2018) and ELMO (Peters et al. 2018), have been developed. These models are undoubtedly better at whole sentence interpretation; since each word does not have a static representation, however, they are not as useful for the semantic interpretation of single words.

LEVERAGING COOCCURRENCE

In response to the success of these models for syntactic and semantic individual word tasks, the tmCooccur action has been developed, which was initially released in SAS® Viya® 3.3. This action computes word cooccurrence statistics in such a way that the results can be factorized using the tmSvd action into a word embedding in a manner similar to the way GloVe and Word2Vec work, except that it can use a sentence context rather than a sliding word context. For natural language, a sentence context might be a better source for semantic interpretation, whereas a sliding word context that doesn't cross sentence boundaries is better for syntactic information. Alternatively, we could use the cooccurrence information directly to form *virtual terms*, or meaningful term combinations. Any of this information, whether to create word embeddings or virtual terms, can be further used to do document-level inference for topics, text categorization, and so on. Finally, text is just one form of sparse, transaction-type data. Theoretically, any of these same techniques can be applied to any type of transactional data for inference and understanding. The following sections detail each of these ideas.

EMBEDDINGS WITH THE TMCOOCCUR ACTION

This section discusses how, in SAS Viya, you can use the tmCooccur action to do word embeddings. Note that this action is inherently parallel and should scale nicely to even very large grids and/or nodes with many processors.

THE TMCOOCCUR ACTION

The tmCooccur action is part of the textUtil (Text Utilities) action set. You can find the documentation for the action in the SAS Viya 3.4 user's guide here: https://go.documentation.sas.com/?docsetId=casvtapg&docsetTarget=cas-textutil-tmcooccur.htm&docsetVersion=8.3&locale=en.

The action first requires you to parse a document collection using any of our parsing approaches to compute an *offset table*, identifying the position in each document a term appears, and a *terms table* to index the terms with unique identifiers. Based on this input, it computes the number of times each term cooccurs with each other term and an *_Association* column that identifies how strongly the terms are connected. The following code is an example call for the data set airlinefeedbackcat, which contains airline feedback comments scraped from a public website:

```
%let sourceloc=\\tmdev\tmine\data\;
libname tm "&sourceloc";

/* Move documents from SAS table to CAS table */
data sascas1.airlinefeedbackcat;
    set tm.airlinefeedbackcat;
    id=_n_;
    run;

/* Parse documents using tmMine action */
proc cas;
loadactionset 'textMining'; run;
action tmMine;
   param
      docid="id"
      documents={ name="airlinefeedbackcat"}
      text="text"
      nounGroups=True
      quittagging=true
      entities="none"
      offset={name="offset", replace=True}
      terms={name="terms", replace=True}
      parseConfig={ name="config",replace=True}
      reduce=8
      stemming=True
      legacyNames=true
;
quit;

/* Run tmCooccur action*/
proc cas;
   loadactionset "textUtil";
   action tmCooccur result=cooccur_res/
      offset={name="offset", caslib="CASUSERHDFS"}
      terms={name="terms", caslib="CASUSERHDFS"}
      cooccurrence={name="cooccur", replace=True}
      cooccurrenceOffset={name="cooccur_pos", replace=True}
      maxDist=25
      minCount=1
      ordered=False
      smoothing=5
      xmax=100
```

```
        frequencyExponent=.6
        distanceExponent=.5
        useParentId=True ;
    run;
quit;
```

Note the process here: First you parse the data, which, depending on options, can optionally include multi-word phrases such as noun groups or entities (such as person name, location, and so on). In addition, if tagging is used, words used as different parts of speech are treated distinctly. Neither of these capabilities are generally present in Word2Vec or GloVe.

Then you take the offset table, which contains every term in every document that is parsed, in order, and the term table and feed those to the tmCooccur action. The action identifies how often each term pair occurs and calculates an association measure based on frequency of cooccurrence compared to expected frequency of cooccurrence.

The parameters used in the example for the tmCooccur action are reasonable choices that our testing found to be generally effective for generating embeddings:

- maxDist=25: This specifies that any word pairs that occur more than 25 words apart in the same sentence will not be treated as a cooccurrence. This is important when sentence boundaries might be missing in some documents. Also, if you want to model more syntactic relationships (we are focused in this paper on semantic relationships primarily), you might want to make it very small, just a few words. In addition, restricting this distance to a value of 3 or 5 will make the results even more similar to GloVe.
- Ordered=false: The tmCooccur action can create statistics for when the first term (or row term) comes earlier in the sentence than the second term (or column term). In that case, set Ordered=true. (See virtual terms later for an example where we do set it that way.) But for purposes of the word embeddings themselves, particularly when you are not using the full term-by-term matrix, set the value to false.
- Mincount=1: The value of 1 indicates every pair of terms is considered since they cooccur at least once. If you have a large collection to analyze, you might want to set the mincount to a higher value.
- useParentID=true: When the value is set to true, all term variations are equivalent to their stems and any synonyms in a synonym list are considered. Setting this value to false will make the result closer to GloVe, which doesn't do stemming.
- Smoothing=5: This setting specifies the alpha smoothing parameter for additive smoothing applied to the counts of the cooccurrences of each term pair (see https://en.wikipedia.org/wiki/Additive_smoothing). If no smoothing is done (smoothing=0), then complete cooccurrence of two rare terms is considered equivalent to complete cooccurrence of two common terms in the correlation calculation, which is unrealistic.
- distanceExponent=.5: This enables you to weight term cooccurrences more strongly when the two words are close together in the sentence than when they are far apart. This value of .5 indicates that terms should be down weighted based on the square root of the number of words between them.

- frequencyExponent=.6: This value indicates how much the term correlation is weighted by the frequency of that cooccurrence to give the association value. Generally, the best results occur when the frequency is raised to an exponent of about .6 or so (very close to the value that GloVe used). Using a frequency exponent greater than 0 guarantees that the most common term cooccurrences have the largest effect on the SVD factorization.
- Xmax=100: Words that are very common frequently cooccur even though their cooccurrences aren't that relevant. This setting controls the maximum frequency that will be raised to that frequency exponent -- any cooccurrence frequencies in excess of this value will be weighted as if it was that value.

These choices worked well for the data we looked at (as described later); of course, your data can differ, and you might want to try some different values to see what works well for you. The nice thing is that the action is quite flexible, with the ability to parameterize it in many ways -- and you might find some useful combinations that we never thought of!

CREATING EMBEDDINGS

Term embeddings are created by applying the SVD to the sparse term-by-term matrix of associations calculated by tmCooccur. In the code here, the tmSvd action performs the matrix factorization and projects the associations into a lower dimensional space of term rows (docpro) and term columns (wordpro). The tmSvd action can rotate these projections into topics, in the same way it does when a term-by-document matrix is fed in. In this example case, we are doing a projection onto 200 dimensions and also rotating the results to create topics:

```
/* Now create word embeddings using tmSvd action for the term by term matrix
created by tmCooccur action */
proc cas;
      loadactionset 'textMining';
      action tmSvd;
      param config="config"
             parent={name="cooccur",where="_Freq2_>=20" }
             termid="_Termid1_"
             docid="_Termid2_"
             count="_Association_"
             terms= "terms"
             timing=True
             rowPivot=0.7
             k=200
             norm="ALL"
             wordpro={name="word_a",replace=True}
             docpro={name="docpro_a",replace=True}
             topics={name="topic_a",replace=True}
             numLabels=7
             topicDecision=False
             u={name="svd_a",replace=True}
             scoreConfig={name="scoreconfig",replace=True}
             legacyNames=True ;
  quit;
```

Note that one issue crops up: When the SVD is calculated on the term-by-document matrix, the wordpro table corresponds to the word projections and the docpro table corresponds to the document projections or the context in which those words occur. But when we do a term-by-term matrix, they are both term projections. Fortunately, in the options for tmCooccur we provided earlier, this matrix is symmetric and both output matrices will be the same (in that case, essentially it would be equivalent to a sparse principal component analysis). However, there is another issue -- the rare term combinations found in the term-by-term matrix will dominate the projections, rendering the results less useful.

To address both issues, specify that you want only the term columns that occur frequently by setting a threshold on the frequency. In the previous code, you can see this in the WHERE clause applied to the parent table where a threshold of 20 is used, so the action is looking at the relationships of every one of the terms with only the most common ones.

The parameters to the tmCooccur action determine how the measure of association is calculated. In general, it is based on a normalized pointwise mutual information score; the result can be considered as a correlation between the terms. That correlation is then optionally weighted by frequency. In fact, with certain parameter choices, the factorization of our association measure would be functionally equivalent to GloVe, if GloVe were able to use sentence context instead of sliding windows. GloVe does a weighted bilinear log regression to calculate the embeddings, as can be seen in Pennington et al. (2014), that is functionally equivalent to an SVD without weighting. What we do is weight the results before doing a straight SVD calculation, which should have a similar result to doing a weighted bilinear log regression.

EVALUATING EMBEDDINGS ON WORD SIMILARITY TASKS

The first question that was addressed was how useful, for semantic interpretation, are the word embeddings generated by this process. To answer this, we downloaded the Billion Word Corpus (http://www.statmt.org/lm-benchmark) and generated word embeddings using the process described in the last section. One of the nice things about the Billion Word Corpus is that all sentences in the English source data are shuffled randomly, with each sentence on a separate line. We also generated word embeddings using the standard SVD approach that SAS® Text Analytics has supported from the beginning, treating each sentence as a separate document.

Finally, we used the gensim Python package to generate embeddings using Word2Vec's skip-gram algorithm on the same Billion Word Corpus (again treating each sentence as a separate document to put it in its best light), and compared all three embeddings, in Python, using the same word similarity tasks as described in the original Mikolov paper. The result is represented as a correlation: The skip-gram embeddings got the best results with a Pearson correlation of .58, the tmCooccur term-by-term embeddings got just slightly below that with a correlation of .56, but the SVD of the term-by-document matrix got by far the worst result with a correlation of .38.

So why did skip-gram get a better result on this task than the tmCooccur generated embeddings? Well, for one thing, the difference is so small it might be just chance. But it might be a real, though slight, difference, in which case we think there are two things to consider:

1. Looking at the word similarity task shows that some of the word similarity is based on syntactic considerations (for example, same part of speech), which would bias them in favor of a technique that uses shorter windows, such as skip-gram, than the entire sentence, which is going to focus on semantic relationships.
2. By treating each sentence as a separate document, we are essentially giving a boost to skip-gram, which typically does not look at sentence boundaries. Here, all word relationships are not considered unless they are within the same sentence, whereas typically skip-gram is not smart enough to know that.

EVALUATING DOCUMENT-LEVEL ENCODINGS

Although the word embeddings themselves are useful for understanding how words relate to each other, for most purposes you are interested in what a document itself is about -- whether it is topic analysis, text categorization, or sentiment analysis. So, the question naturally arises: How can you use these word embeddings to generate document embeddings?

For the standard SAS Text Miner embeddings, which come from an SVD of the term-by-document matrix, the answer is simple. The SVD itself comes out with the document embeddings as one of its factors. Underneath the matrix algebra, each document's embedding is composed as a weighted sum of the term embeddings that are included in that document.

In neither of the other cases is the answer quite so simple -- and in fact, if the document embeddings are composed as simple sums of the term embeddings, the results don't work well. The trick is that in the SAS Text Miner models, the input to the factorization down weights common terms and up weights rare terms. But in the other models, better results are obtained by having the factorization affected positively by higher frequency cooccurrences. In general, it appears that rare terms are better for indicating document similarity, whereas term embeddings themselves seem to benefit from looking at the most common co-occurrences.

The solution, for both skip-gram embeddings and embeddings based on the term-by-term matrix, is to do the weighting after the term embeddings are calculated -- essentially each term is weighted by its inverse document frequency (idf) weighting applied to each embedding dimension. Then the results are fed to the tmScore action to calculate document projections.

Three data sets that had natural text categories already applied were used to evaluate the embeddings:

1. the airline feedback data set obtained by scraping a website containing airline reviews (the categorical variable was the category of the feedback obtained)
2. the 20 Newsgroup data set that has long been a staple of text categorization research
3. the NHTSA vehicle complaint data set from the first quarter of 2008 (the categorical variable was the component of the car that was being commented about)

For each data set, an autotuning neural network (autotune.tuneNeuralNet CAS action) with 10-fold cross validation was used to get an overall accuracy number. In all cases, 200 embedding dimensions were used.

The results are in

Table 1. Accuracy by Model and Data Set

			Models-Accuracy				
data	# docs	# cats	baseline	T-by-D SVD	skip-gram	Sentence T-by-T SVD	Document T-by-T SVD
Airline Feedback	6163	15	32.70%	80.5%	65.9%	73.8%	74.2%
20 Newsgroups	19287	20	5.10%	74.1%	70.3%	76.6%	78.2%
NHTSA 2008	38072	26	12.20%	73.6%	64.8%	73.1%	73.4%

Table 1. Accuracy by Model and Data Set

The table here includes the number of documents in each data set, the number of categories, and a baseline probability (in this case, the percentage of documents in the most common category). The final four columns show the accuracy obtained using the following techniques:

1. T-by-D SVD - A term-by-document SVD that is typically produced by SAS Text Miner
2. skip-gram – A skip-gram model based on a five-word context to both the left and right of a target word
3. Sentence T-by-T SVD - A term-by-term SVD based on tmCooccur
4. Document T-by-T SVD (the final column will be discussed momentarily)

On the 20 Newsgroups data, a slightly better result was obtained for the term-by-term SVD approach that used the co-occurrence of terms in a sentence. For the other two data sets, the best result was for the standard term-by-document factorization. In all three cases, the worst result was obtained from the skip-gram based embeddings.

Perhaps by having a larger context, the term-by-document matrix has more information available to it. One thing you can do is treat every document as if it were a single sentence by setting the sentence variable to a value of one for each observation. This allows the factorization to consider the whole document as the context. In all cases, as seen in the final column, this resulted in a better result for accuracy on this data.

There are likely three reasons the skip-gram model did not do well:

1. It is considering a shorter context and thus has less information available to it.
2. The short context means that embeddings generated from it are a mix of syntactic and semantic information, but the syntactic information is not useful for text categorization.
3. It has no notion of sentence boundaries and went across boundaries when deciding what words influenced other words. On the other hand, tmCooccur takes sentence boundaries into consideration.

VIRTUAL TERMS

Another thing that the tmCooccur action can do is to spot strongly cooccurring terms that can provide context to each other. Consider the following examples of sentences:

- I go for a *run* of at least two *miles* every day.
- The *play* has had a very successful *run*.
- He gave the reigning chess champion quite a *run* for his *money*.
- If you would just *shut* the heck *up,* we can deal with this issue.
- Please *shut* the door on the way out.
- I did*n't* really *like* the movie very much.
- I really *like* the movie a lot!

In the first three sentences, three different senses of the noun "run" are used; in the fourth and fifth sentences, the verb particle form "shut up" versus the plain verb "shut" should be distinguished; and in the last two sentences, the use of the "not" term, "didn't", in one sentence and not in the other causes the sentiment to be polar opposite.

In these cases, if you can identify those important word relationships, then you can treat each of these "pairs of words" as a single term. Fortunately, the tmCooccur action gives you the means to do so. A threshold association score is set, and word pairs that exceed that score are identified. Then each such word pair forms a "virtual term"; that is, a virtual term is a word pair that has strong association and is denoted as "*<term 1>…<term 2>*" (for example, "run… mile" or "n't…like"). You can then treat them as ordinary terms and use them for rule building, word embeddings, topic analysis, document categorization, and so on.

So, a natural question is, if you generate embeddings for virtual terms as well as regular terms, can that improve the categorization results reported earlier? For each of the data sets discussed in the previous section, the tmCooccur action was run, this time setting Ordered=True (so that the term "run…mile" would mean that the word "run" preceded the word "mile" in the sentence). A virtual term association cutoff was identified so that the number of virtual terms would equal the number of actual terms. Then term and document embeddings were generated using the standard TM technique (because the tmCooccur embeddings had not shown a clear advantage over the standard approach), and text categorization accuracy was computed. The results are shown in Table 2 (note that the second column repeats the accuracy for the standard embedding approach from Table 1):

Data	Standard w/o virtual terms	Standard w/virtual terms
Airline Feedback	80.5%	92.4%
20 Newsgroups	74.1%	74.2%
NHTSA 2008	73.6%	73.4%

Table 2. Categorization Accuracy

Interestingly, the airline feedback data showed a huge advantage when including virtual terms, whereas, for at least this categorization, there was no real difference between the categorization for the other two tables. We applied this technique to several other data sets and found a number of situations where including virtual terms (particularly for sentiment) was very useful. On the other hand, in some situations they do not appear to give any additional lift. Even in cases where they did not add additional value in terms of accuracy, they did add descriptive value by including virtual terms in topic labels or by doing automatic rule generation and having the rules include virtual terms.

APPLYING EMBEDDINGS TO NONTEXTUAL DATA

Although the techniques for word embeddings described in this paper are designed to analyze textual data, they are applicable to other types of sequential data. If you think of a sequence of words in a sentence as being an ordered, time-dependent transition from one word to the next, then other forms of time series data also satisfy this situation. Such examples include information about the location of people, vehicles, or robots at specific times of the day, state information of a complex system as it transitions through time, web click data that records a user's path as she traverses a website, and transactional purchase data.

THE AMERICAN TIME USE SURVEY

The American Time Use Survey from the Bureau of Labor Statistics (https://www.bls.gov/tus/) has characteristics that provide a good example. In this survey, users provided information about the various activities they participate in throughout the day such as eating, household activities, working, and driving. Responses are tied to a given person on a given day and are ordered based on the time and duration that they occurred. In this data, there are over 100,000 daily records and a total of over 500 distinct activities recorded.

By merging data files for individuals and activity names, you can see an example of the start of a user's morning in Table 3.

Obs	tucaseid	tustarttim	tustoptime	tuactivity
52	30100014165	04:00:00	05:00:00	Sleeping
53	30100014165	05:00:00	05:30:00	Washing, dressing and grooming oneself
54	30100014165	05:30:00	05:45:00	Travel related to working
55	30100014165	05:45:00	13:00:00	Work, main job

Table 3. Morning Activities for One Individual

Notice how the form mimics the term-by-document compressed frequency table for text. In this setting, an individual's day of activities corresponds to the sentence and each activity corresponds to a word. If you are interested in the cooccurrence of different activities that a person engages in, then you can use the tmCooccur action to evaluate the structure of these co-behaviors -- they function in the same way that the position of a word in a sentence does.

ASSOCIATIONS AND EMBEDDINGS

The tmCooccur action itself provides useful information about the transitions that occur overall in the day-to-day activities. When tmCooccur generates associations with this data, this correlation plot of the strongest correlations shows what activities tend to occur near one another when they do occur.

In Figure 1, the six strongest associations are shown on the diagonals. The off diagonals indicate the association of the activities with the other activities in the plot. The sequential pattern of waiting for services before receiving them was picked up immediately in the association calculation.

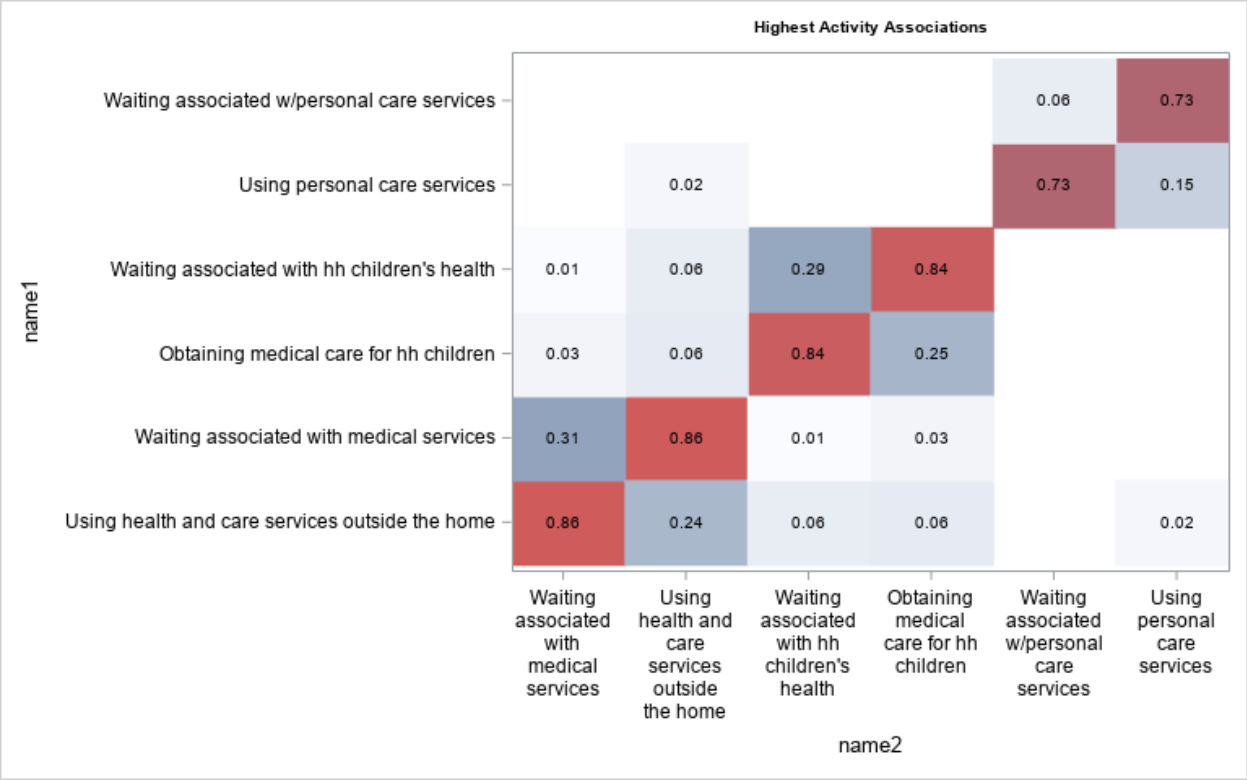

Figure 1. Activities Showing the Strongest Associations

The SVD technique described in this paper can be applied to the associations found from the tmCoccur action. Each activity receives a vector representation derived from the data set. With these embeddings, it is possible to place the terms in a k-dimensional space and check for similarity. Below are three activities and their nearest neighbors in that space. The similarity in this case is based on the context of the individual's day, and not necessarily functionally similar activities.

Activity	Nearest Activity
Sleeplessness	Using in-home health and care services
Interior cleaning	Laundry
Email	Computer use for leisure (exc. Games)

Table 4. Examples of Nearest Activities in the Embedding Space

CONCLUSION

So, as you have seen, context matters. In the case of word similarity tasks based on large corpora, Term cooccurrence embeddings derived from a sentence context can be more effective than those based on the entire document as context. However, when using standard data sources and doing document categorization, using the entire document as context appears to win out. This paradox could be explained by any combination of the following:

1. Sentence-based embeddings might need more data to form meaningful associations than document-based embeddings, based on their use of more information.
2. We might not yet have found the best way to combine sentence-based embeddings to form a document-level embedding.
3. A term-by-document matrix decomposition might just be superior in some way to a term-by-term matrix decomposition.
4. Perhaps there is no natural way to combine word embeddings meaningfully to form a document-level representation.

At any rate, virtual terms appear to be a useful way of disambiguating multiple meanings of terms and, even in some cases when it doesn't increase categorization accuracy, can be useful for descriptive purposes. Future work will entail trying alternative ways of combining term embeddings to form a useful document projection, as well as using context-sensitive embeddings like those provided by ELMO or BERT. Finally, more work will be done to try to apply these techniques to other types of transactional data.

REFERENCES

Firth, John R. 1957. "A Synopsis of Linguistic Theory 1930–1955." *Studies in Linguistic Analysis* 1–32. Oxford: Blackwell.

Devlin, Jacob, et al. 2018. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." CoRR, abs:1810.04805.

Mikolov, Tomas, et al. "Efficient Estimation of Word Representations in Vector Space," CoRR, abs:1301.3781.

Pennington, Jeffrey, et al. "GloVe: Global Vectors for Word Representation." EMNLP. Vol. 14. 2014.

Peters, Matthew E, et al. 2018. "Deep Contextualized Word Representations." *Proceedings of NAACL-HLT 2018*, pages 2227–2237.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the authors:

James A. Cox
SAS Institute Inc.
James.Cox@sas.com

Russell Albright
SAS Institute Inc.
Russell.Albright@sas.com