Reducing the Dimensionality of Data Streams Using Common Sense

Catherine Havasi, Jason Alonso, and **Robert Speer**
MIT Media Lab
20 Ames Street, Cambridge MA 02139
(havasi, jalonso, rspeer)@media.mit.edu

Abstract

Increasingly, we need to computationally understand real-time streams of information in places such as news feeds, speech streams, and social networks. We present Streaming AnalogySpace, an efficient technique that discovers correlations in and makes predictions about sparse natural-language data that arrives in a real-time stream.

AnalogySpace is a noise-resistant PCA-based inference technique designed for use with collaboratively collected common sense knowledge and semantic networks. Streaming AnalogySpace advances this work by computing it incrementally using CCIPCA, and keeping a dense cache of recently-used features to efficiently represent a sparse and open domain. We show that Streaming AnalogySpace converges to the results of standard AnalogySpace, and verify this by evaluating its accuracy empirically on common-sense predictions against standard AnalogySpace.

Introduction

Many approaches to machine learning of semantics are batch methods, requiring that all of their input data be available before they can make conclusions. This is sufficient for some applications, but it is mismatched with many interactive applications, which need to be able to learn on the fly without starting from scratch every time the input is updated.

AnalogySpace is a technique for performing inference over a semantic network, which has the property that it can fill in a node's semantic relations "by analogy" to the relations that other nodes participate in. The existing definition of AnalogySpace, however, requires performing an eigenvector computation over all of the known data at once, making it unwieldy to update this result to take new data into account.

Streaming AnalogySpace is a novel technique that builds on candid covariance-free incremental principal component analysis (CCIPCA) (Weng, Zhang, and Hwang 2003) instead of singular value decomposition (SVD) to discover semantic correlations, enabling its results to be updated incrementally as new data comes in. This enables discovering patterns in and drawing conclusions from semantic data that changes over time, whether it is because the model is continually learning from new examples, or because the state of the world that the data describes is itself changing.

Copyright © 2010, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

AnalogySpace

The Open Mind Common Sense (OMCS) project has been gathering common sense from volunteer users on the Internet, collecting over one million pieces of common sense knowledge to date. This knowledge focuses on the connections between objects in the world, on goals and affectual information, and on events. This user-created resource is designed to form the connections, or glue, that other AI systems need to make sense of relationships in the intuitive way people do (Havasi et al. 2009).

The information in OMCS is processed and released as ConceptNet, a freely available and well supported semantic network (Havasi, Speer, and Alonso 2007). Concepts from OMCS, such as "horse", "take picture", and "sadness", are connected by a set of relations such as "MotivatedByGoal", "PartOf", and "UsedFor". Each of these connections has a strength that increases with the number of OMCS website contributors who approve of each piece of knowledge.

AnalogySpace (Speer, Havasi, and Lieberman 2008) is a noise-resistant principal component analysis (PCA)-based inference algorithm designed for use with common sense information and semantic networks. It was developed as a technique for learning from ConceptNet, but is applicable to other semantic networks as well. In AnalogySpace, a matrix representation of a semantic network is "smoothed" using dimensionality reduction. Broadly, this generalizes from the known information, inferring new connections in strongly-connected areas of the network, and de-emphasizing connections which are poorly supported and likely to be noise.

To create the "canonical" AnalogySpace, which uses ConceptNet as its input data, one expresses the knowledge in ConceptNet as a matrix of concepts and the features that hold true for them, such as "... is part of a house" or "Eating is motivated by ...".

Reducing the dimensionality of this matrix using truncated singular value decomposition has the effect of describing the knowledge in ConceptNet in terms of its most important correlations. The truncated SVD produces a lower-rank approximation of the concept-feature matrix, which has the effect of filling in its gaps with generalizations based on sim-

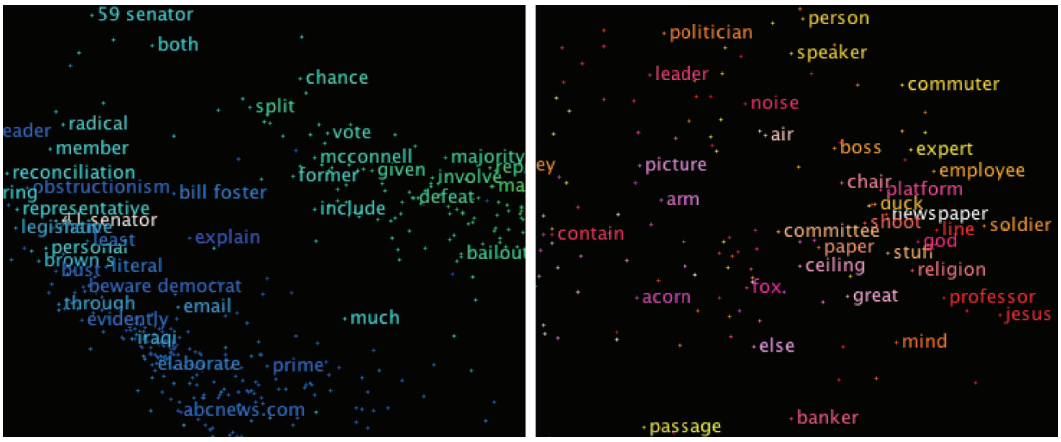

Figure 1: Red Fish Blue Fish displays topics being discussed by liberals (left) and conservatives (right) on January 21, 2010.

ilar concepts and features, and dampening the effect of noisy assertions that are not supported by other similar assertions. It also provides a vector space that represents the concepts and the features simultaneously.

This vector space can be used for inference; this is done by looking up concepts and features that are similar to or associated with a given concept. Thus, concepts and features that point in similar directions, and therefore have high dot products, are good candidates for statements that can be inferred by analogy.

Blending

To increase the applicability of AnalogySpace, one uses a technique called Blending (Havasi et al. 2009) to integrate information from other resources into the SVD decomposition. Blending is a way to perform the AnalogySpace process over multiple data sets, creating a vector space were concepts and features from each resource can be compared. In Blending, before the AnalogySpace process is performed, the multiple data sets are combined into one new matrix, weighted using a parameter called the blending factor (Havasi et al. 2009). This parameter, which can be calculated using a heuristic or specified manually, provides a way to balance the matrices' relative weights in the blend, so that the resulting dimensionality-reduced space takes information from all of them into account. A well-chosen blending factor produces an AnalogySpace that can learn from all the different data sets.

Applying Streaming AnalogySpace to Twitter and Blogs

We currently have two main applications developed using Streaming AnalogySpace: Twittermap and Red Fish Blue Fish. In Twittermap, we examine streams of Twitter messages to create a visual map of the topics currently being discussed. In Red Fish Blue Fish, shown in Figure 1, we analyze the words and phrases appearing on liberal and conservative blogs. The output of Streaming AnalogySpace can be examined using some visualization process such as Landmark MDS (de Silva and Tenenbaum 2004), SVDView (Speer et al. 2010), or a self-organizing map.

In Red Fish Blue Fish, we have applied same principle to political blogs and RSS feeds. This allows us to track different political issues or "loaded words" to see if they align more with liberal or conservative bloggers.

Modifying CCIPCA

To continue evolving the AnalogySpace process, we must create an algorithm based on the same principles but can process continuous information. As our base, we have chosen Candid Covariance-free Incremental Principal Component Analysis (CCIPCA), an incremental PCA algorithm.

CCIPCA is an incremental PCA algorithm that has been shown to be statistically efficient (Weng, Zhang, and Hwang 2003) and has excellent running time and space characteristics. SVD, as well as other PCA and even IPCA methods without the "covariance-free" characteristic, requires the ability to compare every input sample against every other input sample. By selecting a covariance-free algorithm, we only need to keep the table of principal components and an iteration counter. Also of importance, the "candid" characteristic, as described in (Weng, Zhang, and Hwang 2003), reflects that the learning parameters of the algorithm do not need to be tailored to the particular data set to be analyzed.

The Algorithm

The algorithm we give here is slightly different from that published in (Weng, Zhang, and Hwang 2003). We attached a mean-subtraction algorithm that was indicated but not specified in the published algorithm. We also introduced two minor parameters, "remembrance" and "bootstrap", that control when CCIPCA takes effect. We subtly changed the interpretation of the "amnesic" parameter, and we introduced some minor changes for dealing with sparse information. Our variation of the algorithm is given in Figure 2.

Mean-subtraction The original CCIPCA paper assumes that input vectors have zero mean and that "the mean may

Compute the first k dominant eigenvectors, $v_1(n)$, $v_2(n)$, ..., $v_k(n)$, directly from $u(n)$, $n = 1, 2, ...$, given "bootstrap" parameter n_B, "remembrance" parameter n_R, and "amnesic" parameter l.

begin
  $v_0(1) \Leftarrow u(1)$
  for $n \Leftarrow 2, 3, ...$ **do**
    $u_0(n) \Leftarrow u(n)$
    $(w_{old}, w_{new}) \Leftarrow \mathsf{weights}(n, l, n_B, n_R)$
    $v_0(n) \Leftarrow w_{old} v_0(n-1) + w_{new} u_0(n)$
    $u_1(n) \Leftarrow u_0(n) - v_0(n)$
    for $i \Leftarrow 1, 2, ..., \mathsf{min}(k, n)$; **do**
      if $i = n$
        then $v_i(n) \Leftarrow u_i(n)$
        else
          $(w_{old}, w_{new}) \Leftarrow \mathsf{weights}(n, l, n_B, n_R)$
          $v'_i(n) \Leftarrow (u_i(n) \cdot \hat{v}_i(n-1)) u_i(n)$
          $v_i(n) \Leftarrow w_{old} v_i(n-1) + w_{new} v'_i(n)$
          $u_{i+1}(n) \Leftarrow u_i(n) - (u_i(n) \cdot \hat{v}_i(n)) \hat{v}_i(n)$
      fi; **od**; **od**
where funct $\mathsf{weights}(n_{given}, l, n_B, n_R) \equiv$
    $\lceil n \Leftarrow \mathsf{min}(n_{given}, n_R)$
    if $n < n_B$
      then $w_{old} \Leftarrow (n-1)/n$
        $w_{new} \Leftarrow 1/n$
      else $w_{old} \Leftarrow (n-l)/n$
        $w_{new} \Leftarrow l/n$
    fi
    $(w_{old}, w_{new}) \rfloor$
end

Figure 2: The CCIPCA variant used in this paper, aside from the feature queue.

be incrementally estimated and subtracted out." For our purposes, we introduced "eigenvector 0" v_0, which serves as an incrementally-estimated sample mean. For consistency with the rest of the algorithm, we chose to use the same averaging method used in the later PCA stages.

Remembrance and bootstrap We found that CCIPCA fails to converge and produces degenerate results on many randomly-generated vector systems if the published CCIPCA algorithm is allowed to run as originally specified on early iterations ($n \leq l + 1$, in the original interpretation of l). The degenerate results are obvious on inspection of the algorithm: the weights w_{old} and w_{new} have negative and/or zero values during those iterations. We implemented a minimum iteration number, which we call the "bootstrap parameter" n_B, before which the we do not use the amnesic parameter, explained in the next paragraph. We also set an upper limit on the iteration number, which we call the "remembrance parameter" n_R, above which the eigenvectors become moving averages. Though neither of these parameters appear in the original CCIPCA publication, variations of them appear in a publicly-accessible MATLAB file posted by one of the CCIPCA authors (Weng and others 2005).

Amnesic parameter The amnesic parameter controls the degree of influence new samples have over past samples when updating eigenvectors. Our amnesic parameter l is a constant offset from the parameter described in (Weng, Zhang, and Hwang 2003). This gives l an intuitive description: l is the effective number of samples each new input represents for computing the moving average. It should be noted that only the ratio of the amnesic parameter to the remembrance parameter l/n_R impacts the eigenvector learning rate after the iteration number passes the remembrance parameter, when $n \geq n_R$. As such, the amnesic parameter only has meaning when $n_B \leq n < n_R$, and the parameter should only be seen as something to tune to help the eigenvectors converge faster.

Sparse information The input that AnalogySpace works with is generally sparse; of all the possible entries in the concept/feature matrix, most of them are zero. Representing all of these zeros in a dense matrix would require an unreasonably large amount of memory. Standard AnalogySpace deals with this sparsity by using the Lanczos SVD algorithm, which generates a truncated SVD efficiently from sparse input. The sparsity of the input is less of an inherent problem for CCIPCA, because it does not store the entire input at once, but CCIPCA nevertheless converges to a dense matrix that grows continuously as new features appear. To prevent performance loss and unbounded memory requirements, we put an upper bound on the matrix size by using a "feature queue", described further in the next section.

Methodology

In terms of input and output data, Streaming AnalogySpace is roughly the same as standard AnalogySpace. If we create an AnalogySpace from the data in ConceptNet, as is typical for standard AnalogySpace, we would begin with a sparse matrix whose rows represented concepts drawn from ConceptNet (like "a house") and whose columns represented features that could be possessed by those concepts (like "...is located in a neighborhood" or "a garage is part of ..."). In the case of Streaming AnalogySpace, however, we simply present feature vectors to CCIPCA, which then updates its eigenvector table. Once CCIPCA updates its eigenvector table, we use the table to calculate the feature vector's projection into the lower-dimensional space, which is the final AnalogySpace vector.

When updating a CCIPCA, we can obtain inferences that correspond to the inferences that AnalogySpace makes when it reconstructs the A matrix. As a side effect of the update step of CCIPCA, we calculate the correlation of the concept we are updating with each eigenvector. We can then reconstruct the concept as a linear combination of these eigenvectors, producing the same kind of rank-k smoothing as the truncated SVD.

Why run a streaming AnalogySpace over the relatively static data in ConceptNet? One reason is that the data is not entirely static; ConceptNet learns by interacting with people on the Web and asking them questions (Havasi, Speer, and Alonso 2007). Paired with a system that can learn from new knowledge immediately after someone enters it, ConceptNet could increase the interactivity and feedback of its Web interface, by showing the user what new inferences the new knowledge led to, and using those inferences to ask relevant follow-up questions. Even though ConceptNet changes slowly as a whole, the times when it changes are the most interesting times to do inference.

Including other data sources

As described above, Blending is a way of automatically combining different data sources in standard AnalogySpace. In Streaming AnalogySpace, we can perform Blending in a similar way. We alternate which source we are currently accepting new data from, in order to intersperse the different sources of data in time. Thus far, there is no established heuristic for choosing the weights of the different data sources incrementally; instead, we suggest running a small batch SVD over a sample of the data to calculate a blending factor heuristically, and using that same blending factor in Streaming AnalogySpace. In the future, we would like to create a method for automatically tuning the blending factor.

The Feature Queue

The CCIPCA algorithm expresses both concepts and principal components as vectors of features, but particularly in natural language applications, the set of features that can appear is not necessarily known beforehand or even bounded in size. A new feature could appear in the input at any time.

As is the case in standard AnalogySpace, our representation needs to include a mapping from features to vector indices. We keep our CCIPCA variant efficient by using a priority queue to determine which features are mapped to indices. Features that have not appeared in the input for a long time are dropped from the mapping.

The feature queue is a priority queue supporting the *extract-min* operation. The priority of each feature in the queue indicates the last time-step at which it appeared. In this way, we can select the feature to discard simply by extracting the minimum element from the priority queue.

Initially, the feature queue fills up to a specified size, assigning indices to features in order. After the queue is full, every new feature that arrives usurps the index of the least recently used feature. When this happens, the column of the eigenvector matrix representing the contributions of the old feature to each principal component is set to zero.

Equivalence to AnalogySpace

The CCIPCA algorithm is defined to calculate a result that converges to the largest eigenvalues and corresponding eigenvectors of the sample covariance matrix, without having to calculate the covariance matrix itself (Weng, Zhang, and Hwang 2003). The CCIPCA eigenvector matrix V, which can be constructed trivially from vectors $v_1 \ldots v_k$, contains these multiplied together. V can easily be factored into an orthonormal eigenvector matrix, U_c, and a diagonal matrix of eigenvalues, Σ_c.

The singular value decomposition $A = U\Sigma V^T$, meanwhile, calculates U as the eigenvectors of the self-similarity matrix AA^T, and Σ as its eigenvalues. In a truncated SVD, these can be truncated a rank-K product, $A \approx U_k \Sigma_k V^T$, with only the largest k eigenvalues and corresponding eigenvectors represented.

If the input matrix A is normalized to contain unit vectors (as it traditionally is in AnalogySpace), then AA^T will be a covariance matrix. This makes CCIPCA's ideal U_c and the truncated SVD's U_k equivalent to each other. Therefore, given the same input data, U_c will converge to U_k over time.

A difference in the actual implementation of standard AnalogySpace and streaming AnalogySpace, however, is that CCIPCA makes efficient mean subtraction possible, as discussed in the next section. This introduces a difference from standard AnalogySpace, yielding somewhat different results that we will examine empirically.

Improvements on AnalogySpace

The most significant improvements brought by Streaming AnalogySpace are that it can update rapidly according to new data, and that it can adapt to a shifting or unknown problem domain.

Keeping a feature queue (or concept queue), and streaming in the concepts (or features) one at a time, means that we do not need to limit the knowledge we can represent in it to a set that is known in advance. The domain can adapt over time, which is very relevant in open domains such as general knowledge and natural language.

The built-in mean subtraction helps to focus on the inputs that are interesting because they differ from the general trend, even though we may not know from the start what this trend is.

The clearest gain is in its efficiency over streaming data. A batch SVD using Lanczos' algorithm, computing k eigenvectors of an $m \times n$ matrix, has a time complexity of $O(kmn)$ (Zhang et al. 2005). Each step of CCIPCA, using the algorithm on page 2, has a time complexity of $\Theta(km)$. The time required to process n input vectors is still $\Theta(kmn)$, but when new data arrives, it requires only $\Theta(km)$ time to update the matrix. Among other applications, this means that newly-obtained knowledge can update AnalogySpace live and provide immediate feedback, and the existing knowledge does not have to be explicitly reconsidered.

In practical terms, using one core of a 2.8 GHz Mac Pro, our Python/C implementation of Streaming AnalogySpace, with a feature queue of size 10,000 and 50 eigenvectors, requires an average of 36.5 milliseconds per update. For comparison, a 10,000 by 100,000 batch SVD with 50 eigenvectors, using Doug Rohde's SVDLIBC, requires 1.87 seconds per update.

Setting up the CCIPCA is, of course, an investment of time. At this rate of update, it takes around an hour to get an amount of initial data into the CCIPCA that is equivalent to the batch SVD. But this state can be saved, and once it

has been computed it gives the payoff of updates that are 50 times faster.

Related Work

There is very little work applying CCIPCA to natural language processing. In (Li, He, and Zhao 2008), the authors apply CCIPCA to LSA in Chinese, showing that it outperforms dense SVD, though we note that Lanczos-based methods were not considered.

CCIPCA is more often used in graphics and sensor processing. In, (Zhang, Weng, and Zhang 2002), the authors apply CCIPCA to computer vision, wherein CCIPCA is used multiple times in a hierarchical configuration to build a sensory mapping method inspired by early visual pathways in humans. Also on the topic of machine vision, the authors of (Assassa, Mursi, and Aboalsamh 2009) use CCIPCA for face recognition.

Another example of incremental PCA being used in a natural language processing context is in (Gorrell and Webb 2005), where the Generalized Hebbian Algorithm (GHA) is used to do LSA over word and letter bigrams.

GHA (Sanger 1989) and Stochastic Gradient Descent may also be used to accomplish incremental PCA, but CCIPCA outperforms those algorithms in statistical efficiency, given by its rapid convergence, and computational efficiency, given by its computations per iteration (Weng, Zhang, and Hwang 2003). We also observe that our data sets are well-suited to presenting complete vectors for PCA at once, whereas GHA requires processing each matrix entry individually.

Evaluation

Since the goal of a common-sense inference mechanism is to create inferences which are valid and understandable to people, we have chosen human approval as an evaluation mechanism. In the original AnalogySpace paper (Speer, Havasi, and Lieberman 2008) the authors evaluate AnalogySpace by testing people's opinions on the validity of the inferences it produces. This procedure was repeated and here we compared to, amongst other things, the original AnalogySpace which had been evaluated in the previous study.

Our subjects were 78 people of varied ages and genders who completed this evaluation on a webpage. No compensation was offered and none of our subjects knew the sources of our assertions. Each subject viewed 40 assertions, converted from OMCS' internal representation to English sentences using the standard OMCS procedure (Havasi, Speer, and Alonso 2007).

These assertions were created from four sources that were randomly shuffled together, so that participants in our study were doing a blind comparison:

1. 25% of them were existing assertions in ConceptNet, entered by human contributors, sampled randomly from all those with a confidence score of at least 2.
2. 25% were sampled from standard AnalogySpace's predictions about concepts having at least 10 assertions in ConceptNet. This set of predictions includes, for each concept, its top 10 predictions that did not already exist in ConceptNet.
3. 25% were sampled from streaming AnalogySpace's predictions, again from the top 10 predictions for each included concept. The CCIPCA was first trained with a single iteration over the contents of the ConceptNet database, and then these predictions were output on the second iteration.
4. 25% of the assertions were nonsense, generated from random combinations of concepts and features.

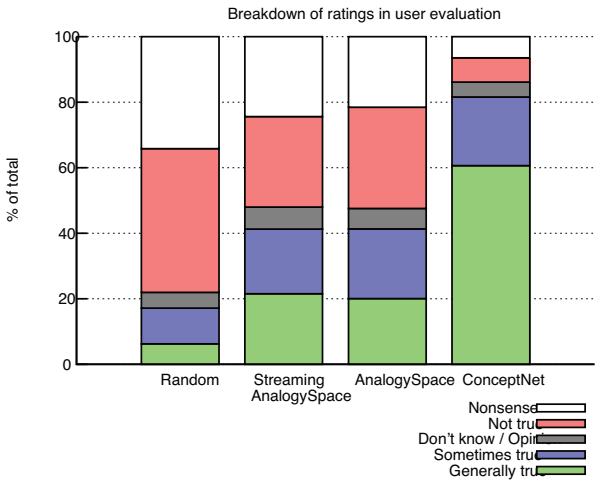

Figure 3: The breakdown of scores that users assigned to statements from the different sources.

Unlike previous evaluations of AnalogySpace, we did not weight the predictions by their score in order to choose higher-scored predictions more often. Because Streaming AnalogySpace made it easy to subtract out the mean of the data, an important PCA operation that was not supported in the existing AnalogySpace code, we could not guarantee that the scores of predictions would be distributed in the same way.

Instead, we sampled *uniformly* from the ten strongest predictions over all concepts. It includes many concepts for which AnalogySpace produces lower-scored, less confident predictions. This greatly expands the domain that is tested, but the expanded domain produces fewer correct inferences than in previous evaluations.

For each assertion, participants selected a choice from the list of "Generally true", "Sometimes / Somewhat true", "Don't know / Opinion", "Generally false", "Doesn't make sense", and "Not true but amusing" (an option suggested by participants in a previous study). These ratings were mapped to a numeric scale as follows: "Generally true" was worth 2, "Sometimes / Somewhat" worth 1, "Don't know / Opinion" worth 0, and all other options were worth -1. The ratings that users gave to statements from each source are shown in Figure 3.

The average results of this evaluation over all 78 participants were:

Source	Mean score	Std. err.
Random	-0.539	0.039
Streaming AnalogySpace	0.110	0.051
Batch AnalogySpace	0.089	0.049
ConceptNet	1.266	0.045

Because different participants in the study apply a different subjective scale when evaluating assertions, we analyze the significance of these results with a correlated-samples one-way ANOVA. The correlated samples are the mean scores assigned to each source by each participant.

The ANOVA shows that there is a highly significant difference across the sources ($F(3, 231) = 331.44, p < .0001$). A Tukey HSD test, giving $HSD[.01] = 0.18$, shows which differences are significant: both inference methods were rated significantly better than random assertions ($p < .01$), and ConceptNet assertions were rated significantly better than the inference methods ($p < .01$), but there was no significant difference in score between the inference methods.

We conclude from this experiment that Streaming AnalogySpace is a reasonable way to infer new connections in a semantic network in practice as well as in theory. After being trained on only two passes through the ConceptNet database, its inferences performed similarly to the existing SVD-based AnalogySpace.

Future Directions

In the future, we plan to explore how we can use this technique to process incoming text to create models in a real-time fashion. We see many applications and research directions in this area.

The AnalogySpace-powered opinion-mining tool Luminoso (Speer et al. 2010) is a tool that uses common sense and blending to better understand opinions and feedback expressed in free text such as customer reviews. It creates a semantic space from the ideas in a set of documents, including common-sense background information, and allows interactive exploration. If Luminoso were combined with Streaming AnalogySpace, we could look at consumer-related information as it is created — during a focus group or chat session. This instant feedback could even guide a group leader during discussion.

As AnalogySpace, Streaming AnalogySpace, and common sense in general were designed to be noise resistant, an interesting future direction of research is to see how Streaming AnalogySpace would work with the output of a speech recognition system. Doing this would enable semantic representations that are grounded in human speech, and could enable new forms of visualization, such as creating an artistic representation of speeches or events as they are going on.

References

Assassa, G. M.; Mursi, M. F. M.; and Aboalsamh, H. A. 2009. Evolutionary eigenspace learning using ccipca and ipca for face recognition. *World Academy of Science, Engineering and Technology* 53.

de Silva, V., and Tenenbaum, J. 2004. Sparse multidimensional scaling using landmark points. Stanford University Technical Report.

Gorrell, G., and Webb, B. 2005. Generalized hebbian algorithm for latent semantic analysis. *Proceedings of Interspeech 2005*.

Havasi, C.; Speer, R.; Pustejovsky, J.; and Lieberman, H. 2009. Digital intuition: Applying common sense using dimensionality reduction. *IEEE Intelligent Systems*.

Havasi, C.; Speer, R.; and Alonso, J. 2007. ConceptNet 3: a flexible, multilingual semantic network for common sense knowledge. In *Recent Advances in Natural Language Processing*.

Li, X.-F.; He, H.-B.; and Zhao, L.-L. 2008. Chinese text categorization based on ccipca and smo. In *Proceedings of the 2008 International Conference on Machine Learning and Cybernetics*.

Sanger, T. 1989. Optimal unsupervised learning in a single-layer linear feedforward neural network. *IEEE Transactions on Neural Networks* 459–473.

Speer, R.; Havasi, C.; Treadway, N.; and Lieberman, H. 2010. Finding your way in a multi-dimensional semantic space with Luminoso. In *Proceedings of Intelligent User Interfaces*.

Speer, R.; Havasi, C.; and Lieberman, H. 2008. AnalogySpace: Reducing the dimensionality of common sense knowledge. *Proceedings of AAAI 2008*.

Weng, J., et al. 2005. ccipca.m. Downloadable source file for MATLAB. `http://www.cse.msu.edu/~weng/research/ccipca.m`, accessed 12-May-2009.

Weng, J.; Zhang, Y.; and Hwang, W.-S. 2003. Candid covariance-free incremental principal component analysis. *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 25(8):1034–1040.

Zhang, S.; Wang, W.; Ford, J.; Makedon, F.; and Pearlman, J. 2005. Using singular value decomposition approximation for collaborative filtering. In *CEC 2005. Seventh IEEE International Conference on E-Commerce Technology*, 257–264.

Zhang, N.; Weng, J.; and Zhang, Z. 2002. A developing sensory mapping for robots. *International Conference on Development and Learning* 13.