Unsupervised Deep Learning Recommender System for Personal Computer Users

Daniel Shapiro* †, Hamza Qassoud*, Mathieu Lemay† and Miodrag Bolic*
*School of Electrical Engineering and Computer Science, University of Ottawa, Ottawa, Ontario, Canada
Email: {dshap092, hqass076, mbolic}@eecs.uottawa.ca
†Clockrr Inc., and Lemay Solutions Consulting Inc., Ottawa, Ontario, Canada
Email: {daniel, matt}@lemaysolutions.com

***Abstract*—This work presents an unsupervised learning approach for training a virtual assistant recommender system, building upon prior work on deep learning neural networks, image processing, mixed-initiative systems, and recommender systems. Intelligent agents can understand the world in intuitive ways with neural networks, and make action recommendations to computer users. The system discussed in this work interprets a computer screen image in order to learn new keywords from the user's screen and associate them to new contexts in a completely unsupervised way, then produce action recommendations to assist the user. It can assist in automating various tasks such as genetics research, computer programming, engaging with social media, and legal research. The action recommendations are personalized to the user, and are produced without integration of the assistant into each individual application executing on the computer. Recommendations can be accepted with a single mouse click by the computer user.**

Keywords–Recommender systems; Unsupervised learning; Deep learning.

I. INTRODUCTION

This work uses a virtual assistant called Automated Virtual Recommendation Agent (AVRA). AVRA follows a Mixed-Initiative (MI) approach to human-computer interaction, where a human and virtual agent work together to achieve common goals. The approach is to offload to AVRA some of the cognitive pressure of understanding onscreen problems and goals visible on the computer screen, and recommending actions to the user as solutions.

AVRA can look into the browser history and screen content history to decipher a simple sequence of events of interest to the user. If the user saw text on the screen in some context and subsequently searched for that text within some small time frame, then AVRA records the event so that it can offer to perform the search when that context and text appear onscreen in the future. In order to decipher which context to train new information into, the AVRA computes with a word embedding model the fit between the new keyword and the keywords already encoded into each context. AVRA also calculates the fit between the screenshot images of the keyword appearing in the user history, and each context already trained into an image processing Convolutional Neural Network (CNN). The image fit is calculated using perceptual hashing [1]. If, for all existing contexts in AVRA, the keyword fit or image fit is too low (as specified by learning hyperparameters), then a new context is created. Training this new context requires obtaining additional training images from the screenshot history, validated using context-specific perceptual hashing. The unsupervised learning algorithm is still in development, and has thus far achieved a 47% accuracy rate in identifying which historical screenshot images are good examples from which to learn the new context.

AVRA learns through the inference of operational knowledge from observations of the computer over time. AVRA monitors the computer screen with regular screen capture images, and analyzes the content of each image to understand what type of visual information is present (the context) and which keywords are on the screen. Each context is associated with a list of keywords, and at any given time there may be multiple onscreen contexts and multiple onscreen keywords. Each keyword in each context is associated with one action. For example, if the Eclipse IDE is present (context=eclipse) and a compiler error is detected in the onscreen text (keyword=NullPointerException), then AVRA can recommend to the user to open a web page containing advice on escaping this error with a try/catch block. In order to offer the most relevant action recommendations, AVRA produces recognition scores for context detection and keyword detection, and combines these scores with a history of user actions to produce an overall score for every possible recommendation.

Several definitions are required in order to discuss unsupervised learning in a compact format. A candidate keyword k is one text snippet within the onscreen text O. The notation $k \in O$ means that the keyword k is in the set O, in this case because it is a substring of O. An action can be referred to as $g \in G$, where g is a particular action and G is the set of all actions known to AVRA. Similarly, a particular context c is one element in a set of contexts learned into AVRA, denoted as $c \in C$. The set of all keywords learned by AVRA is K, and after discovering $k \in O$, AVRA can integrate k to become $k \in K$.

The challenge discussed in this work is to learn new contexts, keywords, and actions without human intervention. How can AVRA autonomously learn new visual contexts into C beyond "eclipse", and new context-specific keywords such as `NullPointerException` into K? Even if learning new contexts and keywords was accomplished, how can AVRA learn which actions G are associated with contexts and keywords? More formally, this unsupervised learning challenge is to:

(TASK 1) Identify keyword k within onscreen text O leading to action g in context c.
(TASK 2) Recognize context c if it appears again.
(TASK 3) Recognize keyword k if it appears again in onscreen text O when context c is recognized.
(TASK 4) Enable AVRA to recommend action g when context c and keyword k are recognized onscreen at the same time.

This work describes a novel approach to unsupervised

 ISBN: 978-1-61208-576-0

learning for a computer assistant. Once AVRA can watch users and draw causal relationships from user actions, it can learn new information unforeseen by its developers. The first step on the path to a working solution was to relax the constraints on the problem and solve easier problems of unsupervised action learning without context learning (Sections III) and supervised context learning (Section IV). The solutions to these simplified problems are then combined and expanded upon in Section V to answer the larger question of how to learn new actions, contexts and keywords. Next, performance experiments are detailed in Section VI. The contribution of this work is to describe how unsupervised learning can be used in a recommender system. Unsupervised action learning without context is discussed in Section III. The methods of unsupervised action learning with supervised context learning are described in Section IV. Section V describes how AVRA can use both unsupervised action learning and unsupervised context learning. Section VI contains a summary of this work and a discussion of future research directions.

II. Prior Art

Deep learning includes three methods for learning: supervised, reinforcement, and unsupervised learning [2]. Supervised Learning (SL) is the best understood and involves training the classifier using many labelled examples. For example, images of cars accompanied by the label "car", alongside images of dogs accompanied by the label "dog" can be used to train a classifier to discriminate between images of cars and dogs. In supervised learning the classifier adjusts weights during each training iteration of processing the dataset in order to minimize the classification error. Unlike supervised learning, reinforcement learning involves training without an immediate reward signal [3][4]. Reinforcement learning is useful in use cases such as autonomous driving cars and strategy games, where the feedback to the learning system only arrives after some end state is reached, or after a significant delay. And finally, Unsupervised Learning (UL) is the process of learning without labelled examples organized into a dataset [5]. This form of learning gets no feedback, and therefore requires that the learner figure out a pattern from raw data and also figure out a metric for evaluating the accuracy of what was learned. In terms of information content, as described in [6], reinforcement learning predicts only a few bits per input sample (e.g., position of steering wheel to control car), supervised learning predicts a few thousand bits (e.g., class labels to add to an image), and finally unsupervised learning predicts anything to do with the input (e.g., given a video, predict images of the next few frames [7]).

Transfer learning is an approach in ML where the training data is augmented by some other already trained model [8, page 243]. The advantage of using transfer learning is that it enables a model to start from some already trained set of learned features and extend this initial set of knowledge by training on additional data, rather than randomly initializing the weights and training from that random starting point. Transfer learning was accomplished in this work using an Inception v3 tensorflow CNN model that was trained on ImageNet images [9]–[11]. The model was extended by training a new last neural network layer on top of the existing fixed network that can recognize new classes of images after training. This final layer of the CNN received a 2048-dimensional input vector for each image, after which a softmax layer is added. As explained in [12], for N labels this CNN learns only N + 2048*N model parameters corresponding to the learned biases and weights. This is a vast decrease in the number of parameters to learn over training all layers of the model.

The training effectiveness of supervised learning can be enhanced by injecting random noise into the inputs to each layer of the neural network during the training process [13] [14], and by randomly dropping out inputs in each layer of the neural network (dropout) [15]. Dropout and random noise injection each help to prevent the model from overfitting the data during training.

Feature engineering is the process of applying domain knowledge and human data analysis to strengthen predictive models such as neural networks [16][17]. The idea of feature engineering is to reorganize existing data into new formats that the learning system can learn more easily. For example, humans perceive the direction "walk straight" more effectively than "walk in the ventral direction orthogonal to the plane formed by the corners of your torso that faces out from your eyes". These two statements contain the same information, but presenting this data in an easy to process format makes all the difference. In feature engineering that can mean re-encoding the data between acquisition and training.

Supervised learning can be thought of as a clustering problem, where a classifier must learn a linear boundary function between two sets of labeled points on a 2 dimensional graph. In reality, this graph could be of higher dimension, the classifier function could be nonlinear, and the graph could contain more than 2 classes, but the idea serves to illustrate the point of what a SL classifier is doing. It learns a classification function based upon labeled data in order to be able to classify novel data. Unsupervised learning can be thought of as a similar clustering problem, with all of the points having no labels. The main difference here is that in the unsupervised learning case, it is not known to the algorithm which points belong on which side of the line. Worse yet, it is not known ahead of time how many clusters the data should break into.

A good example of an unsupervised learning algorithm is Google News story clustering [18]. The system collects similar stories based on their content, and presents them to the user in an organized way. These stories are organized by their content, rather than by an editor. On a related note, this work uses this same dataset, a 300-dimensional set of approximately 3 million vectors extracted/trained from the Google News dataset of approximately 100 billion words [19].

In this work, unsupervised learning is considered in the domain of Recommender Systems (RS). This means learning new recommendations from unlabeled recordings of computer state and user action data. Unlike reinforcement learning and supervised learning, unlabeled data means that there is no error or reward feedback signal available to create a cost function based upon which the quality of new recommendations can be evaluated. The "right" answer is simply not known to the system. The UL algorithm must make its own decisions about creating image classes and text classes (creating keywords and contexts), and deciding the association between them (what keyword belongs in what context).

Deep neural networks and convolutional neural networks do not contain state information. A given input determinis-

 ISBN: 978-1-61208-576-0

tically results in a given output. Other types of neural nets such as Long Short Term Memory (LSTM) or Recurrent Neural Nets (RNN) contain state information and in addition to containing memory units, the output of the neural network can feed back into the input [20]. RNN do not make the Markov assumption, and even so it is difficult for RNN to encapsulate long-term relationships [21]. In this work, LSTM and RNN were not used, as there was a conscious effort to keep the Markovian assumption in an effort to ease the feasibility of developing an unsupervised learning capability.

Unsupervised learning is often applied to facilitate the success of supervised learning [20, page 14]. Often the unsupervised learning is applied to encode raw data into a form that an SL algorithm can succeed with, where the supervised learning algorithm would not succeed on the raw data. For example, pre-training autoencoder weights prior to applying SL with backpropagation [22], and pre-training RNN [23, page 17].

AVRA's unsupervised learning algorithm finds the similarity of keywords to topics, and the similarity of screen images to learned image features. To learn the relationship between keywords unsupervised, a corpus of online text is extracted and analyzed similar to the approaches of [24][25]–[34] and others. To model the semantic relationships between words in the corpus, word embedding is a common approach [24] [30][33]–[36]. Approaches to unsupervised learning applied to semantic word similarity for a corpus obtained from the web include [24] (keyword extraction from spoken documents), [27] (named-entity extraction), [28] (synonym identification), [30] (identifying relationships in a medical corpus), [31] (set expansion), [34] (relation extraction), and [33]. AVRA follows the approach of [33] to iteratively grow topics one keyword at a time based upon the detected context. AVRA also uses the concept of a "Class Vector" introduced in [33] to represent each topic, and the cosine similarity was used in both approaches to measure the distance between vectors. Furthermore, [33] included a crawling mechanism and removed stop words. Named Entity Recognition (NER) was the goal of [33] and the context surrounding the named entity was textual, forming a Bag-of-Context-Words. AVRA instead focuses on keyword recognition (not NER), where the context is visual: what the computer screen looks like when the keyword is detected. Another difference is that [33] focused on different levels of query complexity (Focused, Very Focused, Unfocused) whereas AVRA makes no such distinction between keywords.

III. Unsupervised Action Learning Without Context

Consider a relaxed version of (TASK 1), where there is only one context. The objective is therefore to identify what onscreen text k in the onscreen text O leads to action g. In this relaxed case, (TASK 2) is not necessary, (TASK 3) simplifies to recognizing when text k appears within the onscreen text O, and (TASK 4) simplifies to recommending action g when text k appears onscreen.

Let the input to the unsupervised learning algorithm be the stream of tokenized timestamped Optical Character Recognition (OCR) text $O(t_1)$ produced when processing each computer screen image. Each image is associated with a timestamp t_1. Next, let the actions $g \in G$ be the detected user interest text (e.g., browser search, clipboard history, keystrokes). Each element g in G is an action performed by the user which could conceivably be replayed by AVRA on behalf of the user. Each observed user action is associated with a timestamp t_2, yielding a stream of timestamped actions $G(t_2)$ and corresponding keyword $K(t_2)$. A dictionary F can store the relationship between keywords and actions as $F < k, g >$ allowing AVRA to identify the desired action g when it detects keyword k.

The learning algorithm can iterate through the OCR text $O(t_1)$ and actions $K(t_2)$, and store into F wherever $K(t_2)$ came soon after the OCR text $O(t_1)$ appeared onscreen, and the user interest text $K(t_2)$ was a substring of the onscreen text $O(t_1)$. These constraints are expressed as $[t_2 > t_1]$ *and* $[K(t_2)$ *in* $O(t_1)]$ *and* $[t_2 - t_1 < windowSize]$. Following this approach, the algorithm can learn from scenarios where the user copies onscreen text (a substring of $O(t_1)$) and pastes into a search engine producing the action text $K(t_2)$ for action $G(t_2)$. After learning this pattern in F, AVRA can recommend the relevant action when a keyword appears onscreen, without the user copying and pasting and searching. The algorithm of Figure 1 implements these concepts. It provides a method for determining what onscreen text O leads to action G in this single context problem. The approach is to search for an onscreen $keyword$ that the user searched for in the past (verbatim) after seeing it on the screen.

Input: OCR text of computer screen at time $t1$: $O(t1)$;
Detected user interest text (e.g., browser search, clipboard history, keystrokes) at time $t2$: $G(t2)$
Output: Database of problem / solution pairs
$F < O, G >$

```
for each O(t1) in history do
    for each K(t1) in O(t1) do
        for each G(t2) in history do
            if t2 > t1 and G(t2) in K(t1) and
            t2 − t1 < windowSize then
                F.store(K(t1), G(t2))
            end
        end
    end
end
```

Figure 1. Learning Algorithm: What onscreen text O leads to action G

Consider the example where at time $t = 0$, while AVRA is running in the background, the user and AVRA see an image of a dog and the onscreen word dog. Next, at time $t = 1$, some other information is seen on the screen, and finally at time $t = 2$, the user searches for the word "dog" and AVRA stores into F the fact that the recently observed onscreen word "dog" led to the user performing a search action for that word.

IV. Supervised Context Learning

The challenge in training the CNN with supervised learning is acquiring many images that look like a particular context, in order to carry out the supervised context learning. At least 30 images representing the context should be used to train the CNN in order to avoid total failure of the training. However, 300 to 800 training images is a "good" image dataset size for each context. Only when a sufficient number of images have been collected can the images be used to train the new context into the CNN, or reinforce an existing context with new information. Several image collection approaches are possible:

 ISBN: 978-1-61208-576-0

(METHOD 1) Capture an image representative of the context each time AVRA learns a new keyword into F.
(METHOD 2) Collect images from image search engines based upon context-specific keywords.
(METHOD 3) Given one or more images representing a context, collect additional images using reverse image search.
(METHOD 4) Leverage collaborative filtering to collect context-specific images identified by other AVRA users.

For (METHOD 1), collecting the context training images locally with AVRA was accomplished by simply retaining the screen captures stored by AVRA during routine operation. Working backward from the time when K stored a new keyword, the image captured at time t_1 when k appeared onscreen, the image captured at t_1 should be a picture containing the context of interest C_i. The downside of this approach is that it requires many observations to collect sufficient data to train the CNN. This approach was further improved by sampling the images just before and after t_1 and including them in the training data if they were similar to the image taken at t_1. Similarity was established with a perceptual hash comparing the image taken when the keyword was onscreen, and the images taken at nearby timestamps. For example, if the image taken at time t_1 is a picture of the Eclipse IDE showing a stacktrace containing `NullPointerException`, then the next image taken is likely also a picture of the Eclipse IDE. If these images are in fact similar, then the difference in perceptual hash values between the image taken at time t_1 and the image taken at time image t_{1+1} would be small. Similarly the image taken at time t_{1-1} may be a useful training example if the perceptual hash difference from the image taken at t_1 is small. Image similarity can be controlled by tuning an image similarity hyperparameter.

(METHOD 2), scraping representative images from the Internet to form training datasets, was implemented in nodejs. The program cycled through a hand-crafted list of keywords based upon K relating to the desired context (e.g., "eclipse IDE java programming") and submitted these keywords to image search engine APIs. The search engine submissions returned lists of URLs for images and additional information about these images such as image type and size. The image search was narrowed to include only large images with specific image formats. The next step involved manual data validation where non-representative images were deleted by a human operator. A further step of duplicate image deletion was accomplished with an automated tool.

Whereas (METHOD 2) relied on keyword-based search engines, (METHOD 3) involved querying perceptual hash search engines (e.g., TinEye [37] and Yandex [38]). To find novel images related to the already collected image(s), the perceptual hash of the known image can be used to identify similar images. The downside of this approach was that there may be no such images available, or the identified images may be copies of the submitted image with tiny modifications (e.g., added or modified text).

For (METHOD 4), the collaborative filtering of user actions in a distributed framework with many clients, requiring several instances of an action to be observed before learning a pattern is a reasonable expectation. It is the basis of the collaborative filtering concept that data for one user can be applied to another user.

To learn new contexts in an unsupervised fashion, one or more of these approaches must be automated, removing all human intervention. For example, with (METHOD 2) image search keywords are produced manually, and images are validated manually. With (METHOD 1) the user must perform the same action many times, and the image similarity metric must be flexible enough to allow differences between images but strict enough to reject irrelevant images from polluting the training data.

V. Unsupervised Context Learning

Having outlined solutions to unsupervised action learning and supervised context learning, enough of the solution is revealed that one can begin to consider the full scope of the unsupervised learning problem. Consider AVRA's design shown in Figure 2. How can AVRA autonomously identify keywords K within onscreen text O leading to action G in context C?

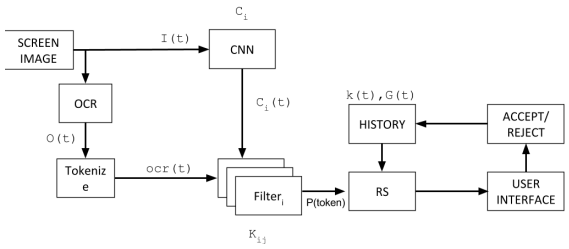

Figure 2. AVRA System Overview.

Consider that AVRA has just detected that a keyword k captured at timestamp t_1 and recognized in image $I(t_1)$ was followed by user action $g(t_2)$. AVRA must decide if this keyword belongs to an existing context or a new one. New challenges emerge when attacking this broader problem definition. First, the fit between a new image $I(t_1)$ and existing trained CNN context C is required to understand how well the new image fits into the set of features that define each context. Second, the relative fit between keyword k and every existing context in C must be quantified in order to decide into which context new information should be stored. Third, AVRA requires an autonomous method for extrapolating novel images from the set of acquired images, as described previously in Section IV.

To obtain the image 'fit', the CNN can tell the unsupervised learning algorithm how much a new image 'looks like' the contexts it was already trained to recognize simply by processing the image in the same way as AVRA interprets screenshots. This capability is exposed by simply processing the image $I(t_1)$ through the CNN and observing the classification confidence score for each context. The output of the CNN indicates how much the image looks like each context.

A trained word embedding model should contain a representation that encodes the semantic understanding of words. The vectors for words can be manipulated to compare ideas, as previously described in the famous $king - man + woman = queen$ example [39]. To find the fit of a new keyword k with an existing context C_i, one or more trained word embedding models are interrogated to find out if k and many keywords of interest K are represented in the model. If so, the cosine

similarity between the keywords K already in the context C_i and the new keyword k is computed. More specifically, AVRA's unsupervised learning algorithm relies on the Google News word embedding model to obtain the conceptual distance between words [19]. If a word is not found in the word embedding model, then the distance is set to 0. The average similarity between k and the keywords in context C_i represents the relative fit of keyword k into C_i. If each context contains n keywords, and there are m contexts trained into AVRA, then keyword k must be compared to $m * n$ elements.

At this point the 'fit' between a new keyword and each context is computed as the similarity between a keyword and each keyword in each context. Each of the $m * n$ calculations peers into the Google News word2vec model, requiring several hours to execute. To accelerate the comparison to several seconds, an average vector for all the keywords in each context is computed, and then compared with the candidate keyword. Several approaches are well known for computing a vector to represent a set of words in a word embedding model, including the average vector approach implemented in AVRA [40], and k-means clustering [8, page 5]. One major advantage of this new approach with average vectors is that computing the average vectors is accomplished outside the word2vec model, and so it executes very quickly. Further complicating matters, some keywords AVRA learned during supervised learning are not available (not trained into) in the word2vec model, and so the vector for those keywords in the model does not exist. Therefore, the fit between the candidate keyword and those missing vectors were not taken into account in creating the average vector. To fix this, the fit between the average vector and the vector for the new keyword k is multiplied by a ratio of A (the number of vectors used to make the average vector) and B (the total number of keywords in the context of interest). And so if all the context keywords are in the model, the ratio is 1.0, if none are, then the ratio is 0.0, and if half are in the model, then the ratio is 0.5. The calculation of the mean similarity is computed as the similarity between k and the average vector, multiplied by the ratio.

Another acceleration technique used was memoization. Any intermediate results (e.g., distance('paris','france')) is not re-calculated when the result is needed later on.

As discussed previously, approaches to getting more images representative of a new CNN context requires fully automating one or more of (METHOD 1) (making multiple observations of $k + C_i \rightarrow g$ before learning a new context, and obtaining similar images nearby in time using perceptual hashing), (METHOD 2) (keyword-based image search engines), (METHOD 3) (perceptual-hash reverse image search), and (METHOD 4) (collaborative filtering). (METHOD 1) was already fully automated, and provides a small number of useful images as a starting point for the CNN training dataset. To automate (METHOD 2), images obtained from (METHOD 1) were fed to an image labeling API (Google Vision API [41]) in order to come up with a set of keywords, and these keywords were submitted to image search engines to obtain new image examples. The resulting images were a poor fit for the contexts tested (e.g., console, Eclipse IDE) as a result of the small number of labels that the image annotation API was able to extract from the available images. Obtaining keywords from images was possible, and scraping images based on these keywords was also possible, but the quality of the generated keywords was too low to be useful for an autonomous use case. For example, the labels added to an image of a console window were: Text, Font, Brand, Screenshot, Design, Presentation, Line, and Document. Searching for images using these labels does not return additional images of console windows. (METHOD 2) automation was therefore not successful. Surprisingly, full automation of (METHOD 3) was similarly disappointing. Reverse image search did sometimes return novel examples of the submitted context (e.g., image of a console window returned additional examples of console windows). However, reverse image search tended to return either no results (e.g., a fullscreen image of the Eclipse IDE) identical copies of the submitted image (e.g., an image of a celebrity) or results focusing on the "wrong" features of the submitted image (e.g., for a screenshot of a desktop, the perceptual hash caused the results to be the image shown as the desktop background with the desktop icons removed, focusing on the irrelevant background image at the expense of the real goad of finding images of desktop backgrounds). Full automation of (METHOD 3) was therefore not successful. Collaborative filtering (METHOD 4) was not implemented.

Having described how the text fit and image fit are computed, and how a dataset to train the CNN can be obtained for new contexts, the unsupervised learning process can now be discussed in additional detail. For each new keyword k under consideration, if AVRA has seen enough examples of the keyword (and related images for context training) triggering an action, then AVRA will begin assessing which context the keyword belongs in. This may be a new context or an existing context. This keyword may already exist in one context and now also belongs in another. Let $I_{examples}$ be a list of the images related to one keyword k. AVRA begins by assessing the keyword in relation to the keywords already learned for each context C_i. The average fit between the new keyword k and the existing keywords in C_i is computed using the ratio with average vector comparison approach described above. Next, the average fit between the images $I_{examples}$ and the context is computed by processing them through the CNN and averaging the classification confidence for class C_i. If $I_{examples}$ does indeed contain similar features to the already trained CNN class, then a high average fit is expected. To associate the keyword to an existing class, the average image fit must exceed hyperparameter h_1 and the keyword fit must exceed hyperparameter h_2, and the average image fit multiplied by the keyword fit must exceed any previously encountered "best context fit". In other words, the class with the strongest keyword and image similarity is assigned the keyword unless either the keyword or images are too dissimilar from any existing context. In that case a new context is learned.

If the keyword is learned into an existing context C_i, then the CNN can be retrained with images $I_{examples}$, and the keyword identification system for C_i is also updated to recognize the new keyword. If, however, the keyword is learned into a new context, then AVRA finds additional distinct images according to (METHOD 1), in an attempt to increase the number of images available for training. This larger image set is used to retrain the CNN to identify the new context. The keyword identification system is also updated to recognize the new keyword.

AVRA's unsupervised learning algorithm described in this Section is presented in the algorithm of Figure 3. Similar to

Figure 1, it can learn causal relationships between onscreen keywords and user actions. However, the added advantage in Figure 3 is that it can also learn features from what the screen looks like when the keyword is present (image contexts). On the first line of Figure 3, the observations made by AVRA are processed to identify a set of keywords ($new\_keywords$) that appeared onscreen prior to the user performing a related search. This part of the algorithm work as described in Figure 1. For each keyword k in $new\_keywords$, a list of images $I_{examples}[k]$ is also collected. Next, for each keyword and corresponding action ($[k, g]$), if there were enough causation examples observed, the algorithm checks each context to see which has the highest context fit (lines 6 to 14). If a best context is found, then the keyword k and action g are added to AVRA's database.

This method for unsupervised learning can be viewed as partitioning the space of all images and words into sub-regions by context and keyword. Keyword clustering is one part of the partitioning, and image clustering is the other. The unsupervised learning approach in AVRA incrementally clusters sets of keywords and stereotypes of images. Figure 4 shows the block diagram for AVRA's unsupervised learning approach. A sufficient number of new keyword identifications (TASK 1) causes a decision engine to assesses a keyword $k(t)$ recognized in image $I(t)$. Next, to accomplish context recognition (TASK 2), the fit between the existing CNN contexts and the new image is computed (Context Similarity in Figure 4). Further to (TASK 2), the images provided to characterize the potential new context are extended by testing the images taken just before and after time t with a perceptual hash, and keeping images with a difference less than $h3$ from $I(t)$. The resulting set of images is called $I_{examples}$. The context similarity task returns the list of contexts sufficiently similar to the potentially new context (with a similarity threshold of $h1$). DNN training to recognize a new keyword within a context (TASK 3) is only initiated once a keyword has been assigned a context. To assign a keyword to a context (new or existing), the keyword fit is first computed (Keyword Clustering in Figure 4), and a list of contexts with sufficiently similar keywords is returned. The text similarity threshold is hyperparameter $h2$. If no context has a sufficiently high keyword fit and context fit, n new context is present, and the CNN is retrained to recognize the new context. However, if there are contexts with sufficiently high keyword fit and context fit, the context with the highest combination of context and keyword fit (computed by multiplying them together) is assigned the new $keyword\ k$. When the keyword is assigned a context, in addition to the DNN training being initiated, action g is associated to the new keyword k in AVRA's database (TASK 4). To extract user actions from the computer, a browser history program was developed to read out keyword search terms and links from the browser along with visit timestamps and page titles. This information was fed into AVRA's database to form the user action history (G).

The key overlap between AVRA's shallow image processing integration and prior work on fullscreen image processing with a CNN to take decisions (e.g., [4]) is the use of a CNN to process the image of the screen, and then using fully connected layers of a Deep Neural Network (DNN) to make a decision. In the case of AVRA, the DNN output is a recommendation to be ranked based upon supervised or unsupervised learning, whereas in [4] the outputs represent joystick positions learned through reinforcement learning.

Input: Minimum observations of keyword and subsequent action $h0 : 1$; Minimum image fit $h1 : 0.1$; Minimum keyword fit $h2 : 0.1$; Minimum CNN recognition confidence to add new image to training data $h3 : 0.1$; Maximum perceptual hash difference to add new image to training data $h4 : 35$; Detected user search text, URL, and timestamp recorded at time $t2 : G(t2)$; Snippets of onscreen text observed at time $t : ocr(t)$; Maximum time from observation of keyword to action by user $windowSize : 10\ s$; List of CNN contexts $contexts$

Output: Keyword added to new context or existing context, or nothing learned.

```
1  [I_examples, new_keywords] =
   detectNewKeywords(G, ocr, windowSize)
2  for each [k, g] in new_keywords do
3      if length(I_examples[k]) ≥ h0 then
4          best_context_fit = 0
5          best_context = None
6          for each context in contexts do
7              keyword_fit =
               modelTextFit(k, context.keywords())
8              img_fit =
               average(CNN_classify(I_examples[k],
9                context))
10             if img_fit > h1 and keyword_fit >
               h2 and img_fit * keyword_fit >
               best_context_fit then
11                 best_context = context
12                 best_context_fit =
                   img_fit * keyword_fit
13             end
14         end
15         if best_context then
16             train_DNN(best_context, k, g)
17             train_CNN(best_context, I_examples[k])
18         end
19         else
20             c = newContextID()
21             I_examples[k] =
               moreImages(I_examples[k], k,
22               CNN_contexts, h3, h4)
23             train_new_DNN(c, k, g)
24             train_CNN(c, I_examples[k])
25         end
26     end
27 end
```

Figure 3. AVRA's unsupervised learning algorithm.

Having described above the unsupervised learning algorithm in AVRA, consider the example of how an existing context can be extended with a new keyword. First, AVRA sees an image of a dog and the text `dog`. The image at timestamp for $t = 0$ is $0.jpg$. At that time, AVRA has already learned through supervised learning two contexts `Animals` and `Colors`. Each context contains two keywords. The `Animals` context contains keywords `cat` and `mule`, while the `Colors` context contains the keywords `red` and `green`. At timestamp $t = 1$, AVRA detects user action g, where the

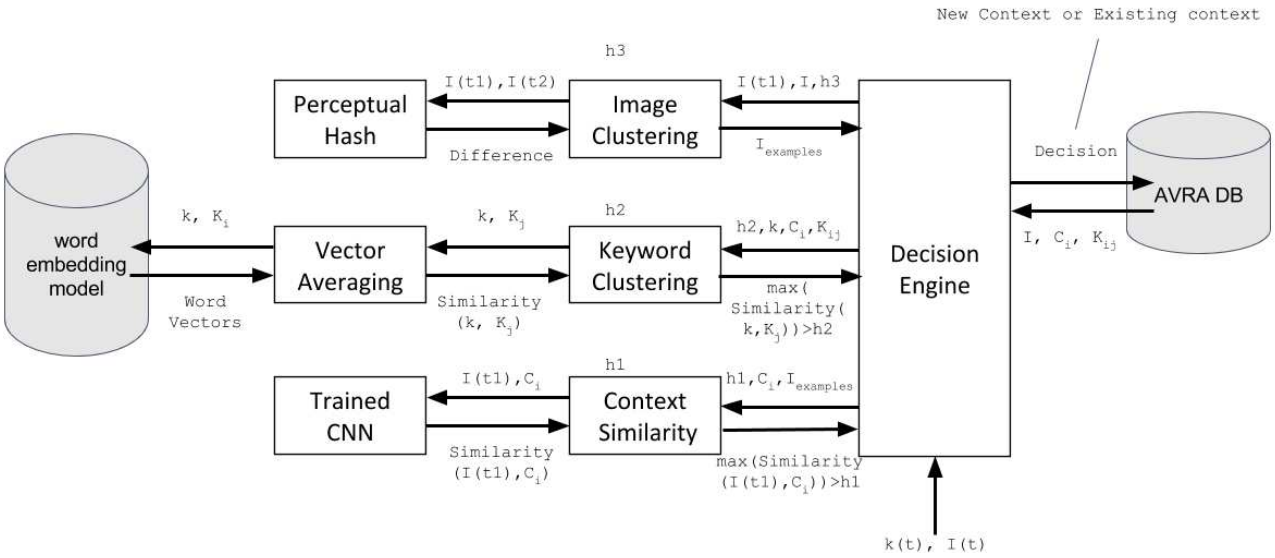

Figure 4. Block diagram for AVRA's unsupervised learning algorithm.

word `dog` is searched for in a browser. The image recorded at timestamp $t = 1$ is $1.jpg$. At time $t = 3$, the unsupervised learning algorithm makes the connection that the text `dog` led to the action `Search(dog)`. The algorithm then loads the hyperparaemters $h1$ and $h2$ as 0.65. Next, the fit of the text `dog` is computed in comparison to the average vector for the context `Animals`, with a result of 0.8. The fit of `dog` with the average vector for the keywords in context `Colors` computes to 0.1. $0.jpg$ is then compared with $1.jpg$ using a perceptual hash to detect if the images are close enough together for $1.jpg$ to be representative of the potential new context. The image difference is too great, and so only $0.jpg$ is used in the next step. The image fit is calculated by passing $0.jpg$ to the CNN for classification. It outputs that $0.jpg$ strongly activates the context `Animals` (0.9 - perhaps detecting the eyes and other body features common to all animals), and also activates the context `Colors` (0.7 - perhaps picking up on the fact that the image of the dog is mostly one solid white color). The ratio for both the `Animals` and `Colors` contexts was 1.0, and so the ratio did not modify the decision at the output in this case. The overall fit of keyword `dog` into context `Animals` was 0.72, exceeding the threshold of 0.65. At 0.07, the overall fit of keyword `dog` into context `Colors` did not exceed the threshold of 0.65, and so it was discarded. With only one context vying to accept the new keyword `dog`, it was added to the context `Animals`.

Consider the second example of AVRA learning a new context. AVRA sees an image of a dog and the text `dog`. The image at timestamp for $t = 0$ is $0.jpg$. At that time AVRA has already learned through supervised learning two contexts `Shapes` and `Colors`. Each context contains two keywords. The `Shapes` context contains keywords `round` and `line`, while the `Colors` context contains the keywords `red` and `green`. At timestamp $t = 1$, AVRA detects user action g, where the word `dog` is searched for in a browser. The image recorded at timestamp $t = 1$ is $1.jpg$. At time $t = 3$ the unsupervised learning algorithm makes the connection that the text `dog` led to the action `Search(dog)`. The algorithm then loads the hyperparaemters $h1$ and $h2$ as 0.65. Next, the fit of the text `dog` is computed in comparison to the average vector for the context `Shapes`, with a result of 0.0. The fit of `dog` with the average vector for the keywords in context `Colors` computes to 0.1. $0.jpg$ is then compared with $1.jpg$ using a perceptual hash to detect if the images are close enough together for $1.jpg$ to be representative of the potential new context. The image difference is too great, and so only $0.jpg$ is used in the next step. The image fit is calculated by passing $0.jpg$ to the CNN for classification. It outputs that $0.jpg$ activates the context `Shapes` (0.6), and also activates the context `Colors` (0.7). The ratio for both the `Shapes` and `Colors` contexts was 1.0 because all of the keywords in each context was used to compose their average vector. The overall fit of keyword `dog` into context `Shapes` was 0.0, below the threshold of 0.65. At 0.07, the overall fit of keyword `dog` into context `Colors` also did not exceed the threshold of 0.65. With no remaining context into which the keyword `dog` can be trained, a new context `newContext` was created and the keyword `dog` was added to it. The arbitrary name `newContext` reflects the fact that AVRA does not know what the overall context is going to store in the future.

A problem surfaced when assessing AVRA's ability to learn new $keywords$ into existing contexts created through supervised learning. The word embedding component of the unsupervised learning algorithm was mostly unsuccessful adding to the supervised learning data. It emerged that the problem was the ratio. The ratio is small when many of the keywords from supervised learning (e.g., nullpointerexception) are not contained in the word embedding model generated from the Google News dataset [19], or other general word embedding models. The model in question contains 3 million words, but this was not sufficient. One approach to force the unsupervised learning model to work was to ignore the ratio, setting it to 1.0 instead of calculating the correct value. Setting ratio to 1.0 is of course a sub-optimal solution, but with this approach AVRA was able to learn new $keywords$ into existing contexts created through supervised learning.

A better approach to learn new $keywords$ into existing contexts created through supervised learning was to re-purpose (METHOD 2) to collect website contents instead of images. The idea was to scrape text from search engine results (web pages), where the search query is built using the keywords AVRA knows about, and the new keyword AVRA wants to classify into a new or existing context. Using the text from

 ISBN: 978-1-61208-576-0

these web pages as a corpus, one can train a new word embedding model that can relate a high ratio of the keywords to each other. The drawback of this approach is that scraping the pages and training the word embedding model is very slow. It can take days. Luckily the unsupervised learning process can be trained as an offline server-side process without slowing down the user experience at all. To build the corpus, the first 30 results for each search term were downloaded. Search terms were composed of distinct sets of 3 or 4 keywords (e.g., $horse\ baseexception\ chicken$). Stopwords were removed using the NLTK Stopwords corpus by Porter [42] [43, page 47]. Stemming was applied using the `PorterStemmer` module of the `gensim` library [44]. Some web pages from the results were skipped because the server refused the connection, the file on the server was not a web page (e.g., PowerPoint file), or the page contained no text. Each valid link returned by the search engine was processed into a text string, and all of the results were concatenated into a single corpus tokenized by spaces. Next, a word embedding model was trained on the corpus, exposing the similarity between words by computing the cosine similarity in the word embedding model between vectors for words.

At this point in the work, it has been described to the reader how AVRA has the ability to make sense of new words by building its own word embedding models completely unsupervised. Crucially, supervised and unsupervised learning can be combined in AVRA, to accomplish transfer learning. AVRA can learn new things autonomously as long as the related keywords are trained into one of the word embedding models that informs the textual context relationships between them, or the information relating these concepts is obtainable by building a word embedding using documents obtained by searching on the web. Multiple word embedding models could be used by AVRA as a general knowledge reference.

VI. Performance Evaluation for Unsupervised Learning

Learning in AVRA is data driven. In this Section, the operation of AVRA's unsupervised learning algorithm with real data is explored. A high-level view of a word embedding model for several contexts is presented to clarify the ability of AVRA to classify new keywords into existing contexts. Visualization for AVRA's image recognition system similarly reveals that AVRA can successfully discriminate between different sets of images. To collect data quickly, a test automation program was used to model a user using the computer (a 'bot'). This bot was used to carry out use cases such as extending an existing context with new information, and creating a new context in AVRA's model. Examples of extending an existing context and creating a new context are provided as validation of the AVRA prototype's ability to apply unsupervised learning.

1) Unsupervised Learning Extending Existing AVRA Context: Table I presents a real example to show how AVRA extends the Eclipse IDE context created using supervised learning. For this example, 5 existing contexts were included in the computations, to give the reader a sense for the computations AVRA performs without overwhelming the reader with many contexts and keywords. Setting the stage for this example, and with AVRA running in the background, a program generated an error, then opened a browser window to search for keywords related to this error message. When AVRA observed the causal relationship between the onscreen error in the IDE, and the search action in the browser, it stored the data in the AVRA database. When sufficient copies of the action were observed, the unsupervised learning algorithm was triggered to try and learn the new keywords into AVRA's RS.

TABLE I. EXTENDING AN EXISTING CONTEXT AFTER OBSERVING THE USER.

Event	Context Similarity	Word Clustering	AVRA Decision	Correct?
New keyword $thread$	$console$ 0.02 **eclipse 0.94** $desktop$ 0.01 $facebook$ 0.02 $gene$ 0.00	**console 0.28** **eclipse 0.15** **desktop 0.30** $facebook$ 0.00 **gene 0.25**	Train $thread$ into $eclipse$	YES
New keyword $exception$	$console$ 0.02 **eclipse 0.94** $desktop$ 0.01 $facebook$ 0.02 $gene$ 0.00	**console 0.16** **eclipse 0.51** $desktop$ 0.01 $facebook$ 0.00 $gene$ 0.01	Train $exception$ into $eclipse$	YES
New keyword $throwing$	$console$ 0.02 **eclipse 0.94** $desktop$ 0.01 $facebook$ 0.02 $gene$ 0.00	**console 0.14** $eclipse$ 0.09 $desktop$ 0.01 $facebook$ 0.00 $gene$ 0.01	Image clustering. Train new context for $throwing$	NO

Examining Table I, AVRA found that images when $thread$ was onscreen strongly activated the $eclipse$ context (0.95) and that word $thread$ was semantically similar to the keywords in $console$ (0.28), $eclipse$ (0.15), $desktop$ (0.30), and $gene$ (0.25). Because only one context demonstrated sufficient image and word similarity, the new keyword $thread$ was trained into AVRA for the $eclipse$ context. Continuing with the second row of Table I, AVRA found that images when $exception$ was onscreen strongly activated the eclipse context (0.94) and that word $exception$ was semantically similar to the keywords in $console$ (0.16), and $eclipse$ (0.51). Because, once again, only one context demonstrated sufficient image and word similarity, the new keyword $exception$ was trained into AVRA for the $eclipse$ context. The unsupervised learning algorithm in AVRA does make mistakes. For example, in the third row of Table I, AVRA recognized the images of the Eclipse IDE but just missed the hyperparameter cutoff of 0.10 to consider the keyword $throwing$ semantically similar to the $eclipse$ context. Instead of learning $throwing$ into $eclipse$, AVRA incorrectly learned the keyword into a new context.

2) Unsupervised Learning Creating New AVRA Context: Table II presents a real example to show how AVRA creates a new context using unsupervised learning. A **bolded** result in the table below indicates a result that exceeded the required threshold. For this example, 5 existing contexts were included in the computations. Prior to the events listed in Table II, the user moved from the a browser window containing a cake recipe to a browser search window and searched for keywords related to the recipe ($chocolate$, $cake$, and $cupcake$). All the while, AVRA was running in the background collecting images and extracting onscreen text. When AVRA observed the causal relationship between the onscreen recipe text, and the search actions in the browser, it stored the data in the AVRA database. When sufficient copies of the action were observed, the unsupervised learning algorithm was triggered to try and learn the new keywords into AVRA's RS.

Starting with the first row of Table II, AVRA found that images of a recipe website taken when the word $cupcake$ was onscreen strongly activated the facebook context (0.90) and that word $cupacke$ was semantically similar to the keywords in $desktop$ (0.18). Because no context demonstrated sufficient

TABLE II. CREATING A NEW CONTEXT AFTER OBSERVING THE USER.

Event	Context Similarity	Word Clustering	AVRA Decision	Correct?
New keyword $cupcake$	$console$ 0.02 $eclipse$ 0.04 $desktop$ 0.03 **facebook 0.90** $gene$ 0.01	$console$ 0.07 $eclipse$ 0.00 **desktop 0.18** $facebook$ 0.00 $gene$ 0.05	Image clustering. Train $cupcake$ into new context	YES
New keyword $chocolate$	$console$ 0.02 $eclipse$ 0.04 $desktop$ 0.03 **facebook 0.90** $gene$ 0.01 **newContext 0.70**	$console$ 0.04 $eclipse$ 0.00 **desktop 0.12** $facebook$ 0.00 $gene$ 0.05 **newContext 0.55**	$newContext$ Train $chocolate$ into $newContext$	YES
New keyword $cake$	$console$ 0.02 $eclipse$ 0.04 $desktop$ 0.03 **facebook 0.90** $gene$ 0.01 **newContext 0.70**	$console$ 0.05 $eclipse$ 0.00 **desktop 0.12** $facebook$ 0.00 $gene$ 0.05 **newContext 0.60**	Train $cake$ into $newContext$	YES

image and word similarity, a new context was created in AVRA. Continuing with the second row of Table II, AVRA found that images captured when the word $chocolate$ was on-screen strongly activated the $facebook$ context (0.90) as well as the new context $newContext$ (0.70). The word $chocolate$ was semantically similar to the keywords in $desktop$ (0.12), and $newContext$ (0.55). Because only one context demonstrated sufficient image and word similarity, the new keyword $chocolate$ was trained into AVRA for the $newContext$ context. For the third row of Table II, AVRA recognized the images of the recipe website, and considered the keyword $cupcake$ semantically similar to the context $newContext$. AVRA learned the keyword $cake$ into the correct context. This example shows that when the risk of concept drift is highest, for a new context with only a few keywords, AVRA does generally manage to build up the new context. There are cases such as the third row in Table I, where keyword clustering or image clustering fails to group an action into an existing context where it belongs, fracturing the context into two (or more) contexts.

3) Unsupervised Learning Relationships Between Keywords: It is interesting to ask how long it takes to train a new word embedding model for a new $keyword$ (e.g., "nullpointerexception") that is not in AVRA's default model, and how well that model works, given the fact that the raw data was built through analyzing web pages returned by a search engine.

To evaluate the ability of the generated model to classify new entities, two small sets of related keywords were created for testing purposes: `Animals` ($horse, dog, cow, pig$) and `Java` ($baseexception, exception, standarderror, importerror$), and the similarity (from the cosine distance) to a new keyword `chicken` was measured. An effective model should find that `chicken` has a lower cosine distance to the average vector for `Animals` than it does compared to the average vector for `Java` keywords.

The $23MB$ corpus of text was downloaded and trained in approximately 30 minutes for 9 keywords. $1,744$ of the links produced usable text. In the collected corpus, the frequency of the stemmed keywords was as follows: $hors(8,137)$, $dog(14,412)$, $cow(4,914)$, $pig(9,933)$, $baseexcept(434)$, $except(9,109)$, $standarderror(256)$, $importerror(544)$, $chicken(6,252)$. The average sentence length was 110.9 characters, and $47,566$ distinct keywords were translated into word vectors in the trained model. The model was created under various hyperparameter configurations (random seed value, training iterations between 5 and 50, context window size between 10 and 40) and each configuration was tested 10 times. All of these measurements resulted in assignment of the new keyword `chicken` to the context `Animals`. Generally, there was a negative similarity between the keyword `chicken` and the context `Java`, while there was always a positive similarity between the keyword `chicken` and the context `Animals`. Very surprisingly, the outcome was positive even when the number of dimensions (also called the number of features) used to represent word vectors was varied between 10 and 100. AVRA sometimes misses the context or keyword information, or has higher confidence in unhelpful recommendations than detected helpful recommendations. Two overall challenges in developing AVRA were poor classification of keywords with very short text length (e.g., the terminal command "ls"), and low context detection confidence (e.g., 2% confidence in the correct class). These cases were rare but noticeable. Perhaps the short keyword recognition could be resolved by modifying the DNN input filter hyperparameters. The low confidence context detection cases may be mitigated by collecting additional image data for context training.

VII. CONCLUSION

This work presented AVRA's unsupervised learning approach and explained with examples how AVRA combined supervised and unsupervised learning to accomplish transfer learning. An architecture for a deep learning recommender system for personal computer users was described in this work. Action recommendations produced by this design are personalized to the user and are generated in real-time. The AVRA system mines information from screen capture data, rather than interfacing with individual applications. Recommendations are presented to the user in an intuitive button-based user interface. The architecture described in this work can provide the foundation for further research into recommender system for personal computer users.

Future work planned for AVRA includes user acceptance testing, testing with large sets of contexts and keywords, collaborative filtering and related privacy considerations, the expansion of AVRA's input processing and modeling capabilities, and more on unsupervised learning. Applying content-based image recognition and semantic segmentation of images to achieve face and object classification within a context (and generating related recommendations) is an interesting area to explore.

REFERENCES

[1] J. Buchner, "A Python Perceptual Image Hashing Module," https://github.com/JohannesBuchner/imagehash, 2017, [retrieved: 2017-05].

[2] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," Nature, vol. 521, no. 7553, 2015, pp. 436–444.

[3] R. Sutton, "Introduction to reinforcement learning," vol. 135, 1998.

[4] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski et al., "Human-level control through deep reinforcement learning," Nature, vol. 518, no. 7540, 2015, pp. 529–533.

[5] G. E. Hinton, P. Dayan, B. J. Frey, and R. M. Neal, "The" wake-sleep" algorithm for unsupervised neural networks," Science, vol. 268, no. 5214, 1995, p. 1158.

 ISBN: 978-1-61208-576-0

[6] Y. Lecun, "Ethics of artificial intelligence - general issues," https://youtu.be/tNWqOgNDnCw?t=1h23m21s, [retrieved: 2017-01].

[7] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," arXiv preprint arXiv:1511.06434, 2015.

[8] E. S. Olivas, J. D. M. Guerrero, M. M. Sober, J. R. M. Benedito, and A. J. S. López, "Handbook of research on machine learning applications and trends: Algorithms, methods, and techniques," 2010.

[9] "Imagenet," http://www.image-net.org/, [retrieved: 2017-01].

[10] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 1–9.

[11] M. Abadi, A. Agarwal, P. Barham et al., "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org [retrieved: 2017-05]. [Online]. Available: http://tensorflow.org/

[12] P. Warden, "Tensorflow for poets," https://petewarden.com/2016/02/28/tensorflow-for-poets/, [retrieved: 2017-01].

[13] W. M. Brown, T. D. Gedeon, and D. I. Groves, "Use of noise to augment training data: a neural network method of mineral–potential mapping in regions of limited known deposit examples," Natural Resources Research, vol. 12, no. 2, 2003, pp. 141–152.

[14] J. Sietsma and R. J. Dow, "Creating artificial neural networks that generalize," Neural networks, vol. 4, no. 1, 1991, pp. 67–79.

[15] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," The Journal of Machine Learning Research, vol. 15, no. 1, 2014, pp. 1929–1958.

[16] L. P. Coelho and W. Richert, Building machine learning systems with Python. Packt Publishing Ltd, 2015.

[17] F. Seide, G. Li, X. Chen, and D. Yu, "Feature engineering in context-dependent deep neural networks for conversational speech transcription," in 2011 IEEE Workshop on Automatic Speech Recognition Understanding, Dec 2011, pp. 24–29.

[18] A. S. Das, M. Datar, A. Garg, and S. Rajaram, "Google news personalization: scalable online collaborative filtering," in Proceedings of the 16th international conference on World Wide Web. ACM, 2007, pp. 271–280.

[19] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in Advances in neural information processing systems, 2013, pp. 3111–3119.

[20] J. Schmidhuber, "Deep learning in neural networks: An overview," Neural Networks, vol. 61, 2015, pp. 85–117.

[21] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," IEEE transactions on neural networks, vol. 5, no. 2, 1994, pp. 157–166.

[22] D. H. Ballard, "Modular learning in neural networks." in AAAI, 1987, pp. 279–284.

[23] J. Schmidhuber, "Learning complex, extended sequences using the principle of history compression," Neural Computation, vol. 4, no. 2, 1992, pp. 234–242.

[24] Y. N. Chen, Y. Huang, H. Y. Lee, and L. S. Lee, "Unsupervised two-stage keyword extraction from spoken documents by topic coherence and support vector machine," in 2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), March 2012, pp. 5041–5044.

[25] G. Grefenstette and L. Muchemi, "Determining the characteristic vocabulary for a specialized dictionary using word2vec and a directed crawler," in GLOBALEX 2016: Lexicographic Resources for Human Language Technology, May 2016.

[26] P. Pantel, E. Crestan, A. Borkovsky, A.-M. Popescu, and V. Vyas, "Web-scale distributional similarity and entity set expansion," in Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2 - Volume 2, ser. EMNLP '09. Stroudsburg, PA, USA: Association for Computational Linguistics, 2009, pp. 938–947. [Online]. Available: http://dl.acm.org/citation.cfm?id=1699571.1699635

[27] O. Etzioni, M. Cafarella, D. Downey, A.-M. Popescu, T. Shaked, S. Soderland, D. S. Weld, and A. Yates, "Unsupervised named-entity extraction from the web: An experimental study," Artificial intelligence, vol. 165, no. 1, 2005, pp. 91–134.

[28] P. D. Turney, "Mining the web for synonyms: Pmi-ir versus lsa on toefl," in European Conference on Machine Learning. Springer, 2001, pp. 491–502.

[29] S. Chakrabarti, M. van den Berg, and B. Dom, "Focused crawling: a new approach to topic-specific web resource discovery," Computer Networks, vol. 31, no. 1116, 1999, pp. 1623 – 1640. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128699000523

[30] J. A. Miñarro-Giménez, O. Marín-Alonso, and M. Samwald, "Applying deep learning techniques on medical corpora from the world wide web: a prototypical system and evaluation," CoRR, vol. abs/1502.03682, 2015, [retrieved: 2017-05]. [Online]. Available: http://arxiv.org/abs/1502.03682

[31] R. C. Wang and W. W. Cohen, "Language-independent set expansion of named entities using the web," in Data Mining, 2007. ICDM 2007. Seventh IEEE International Conference on. IEEE, 2007, pp. 342–350.

[32] R. C. Wang, N. Schlaefer, W. W. Cohen, and E. Nyberg, "Automatic set expansion for list question answering," in Proceedings of the Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics, 2008, pp. 947–954.

[33] A. Alasiry, M. Levene, and A. Poulovassilis, "Mining named entities from search engine query logs," in Proceedings of the 18th International Database Engineering & Applications Symposium, ser. IDEAS '14. New York, NY, USA: ACM, 2014, pp. 46–56. [Online]. Available: http://doi.acm.org/10.1145/2628194.2628224

[34] B. Min, S. Shi, R. Grishman, and C.-Y. Lin, "Towards large-scale unsupervised relation extraction from the web," Int. J. Semant. Web Inf. Syst., vol. 8, no. 3, Jul. 2012, pp. 1–23. [Online]. Available: http://dx.doi.org/10.4018/jswis.2012070101

[35] T. Wang, V. Viswanath, and P. Chen, "Extended topic model for word dependency," in Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and The 7th International Joint Conference of the Asian Federation of Natural Language Processing, 2015, p. 506. [Online]. Available: http://acl2015.org/

[36] W. Li, X. Xie, J. Hu, Z. Zhang, and Y. Zhang, "Using big data from the web to train chinese traffic word representation model in vector space," in 2016 12th World Congress on Intelligent Control and Automation (WCICA), June 2016, pp. 2304–2307.

[37] TinEye, "TinEye Reverse Image Search," https://www.tineye.com/, [retrieved:2017-02].

[38] Yandex, "Yandex.Images: search for images on the internet, search by image," https://yandex.com/images/, [retrieved:2017-02].

[39] T. Mikolov, W.-t. Yih, and G. Zweig, "Linguistic regularities in continuous space word representations." in HLT-NAACL, vol. 13, 2013, pp. 746–751.

[40] Q. Le and T. Mikolov, "Distributed representations of sentences and documents," in Proceedings of the 31st International Conference on Machine Learning (ICML-14), 2014, pp. 1188–1196.

[41] Google Inc., "Vision API - Image Content Analysis — Google Cloud Platform," https://cloud.google.com/vision/, [retrieved: 2017-01].

[42] M. F. Porter, "An algorithm for suffix stripping," Program, vol. 14, no. 3, 1980, pp. 130–137.

[43] S. Bird, E. Klein, and E. Loper, Natural language processing with Python. O'Reilly Media, Inc., 2009.

[44] R. Řehůřek and P. Sojka, "Software Framework for Topic Modelling with Large Corpora," in Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks. Valletta, Malta: ELRA, May 2010, pp. 45–50, http://is.muni.cz/publication/884893/en.

 ISBN: 978-1-61208-576-0