Interactive Visualizations of Word Embeddings for K-12 Students

Saptarashmi Bandyopadhyay[1], Jason Xu[2], Neel Pawar[2], David Touretzky[2]

[1]University of Maryland, College Park, College Park, MD 20742
[2]Carnegie Mellon University, Pittsburgh, PA 15213
sapta.band59@gmail.com, jasonx@andrew.cmu.edu, npawar@andrew.cmu.edu, dst@cs.cmu.edu

Abstract

Word embeddings, which represent words as dense feature vectors, are widely used in natural language processing. In their seminal paper on word2vec, Mikolov and colleagues showed that a feature space created by training a word prediction network on a large text corpus will encode semantic information that supports analogy by vector arithmetic, e.g., "king" minus "man" plus "woman" equals "queen".

We describe a new interactive tool for visually exploring word embeddings. Our tool allows users to define semantic dimensions by specifying opposed word pairs, e.g., gender is defined by pairs such as boy/girl and father/mother, and age by pairs such as father/son and mother/daughter. Words are plotted as points in a zoomable and rotatable 3D space, where the third "residual" dimension encodes distance from the hyperplane defined by all the opposed word vectors with age and gender subtracted out. Our tool allows users to visualize vector analogies, drawing the vector from "king" to "man" and a parallel vector from "woman" to "king-man+woman", which is closest to "queen". Visually browsing the embedding space and experimenting with this tool can make word embeddings more intuitive. We include a series of experiments teachers can use to help K-12 students appreciate the strengths and limitations of this representation.

Introduction

Embeddings are low dimensional representations of points in a higher dimensional vector space. Word embeddings are dense vector representations of *words* in a lower dimensional space, i.e., the number of dimensions is considerably less than the number of words. Word embeddings are capable of capturing both the context in which a word is likely to appear and some aspects of its meaning. Seminal work in this area was done by Mikolov and his colleagues at Google, who created a family of algorithms known as word2vec (Mikolov et al. 2013a,b) that construct embeddings via backpropagation learning. Most new word embedding techniques rely on a neural network architecture instead of more traditional n-gram models and unsupervised learning. The word2vec model of word embeddings has found widespread use in Natural Language Processing (NLP).

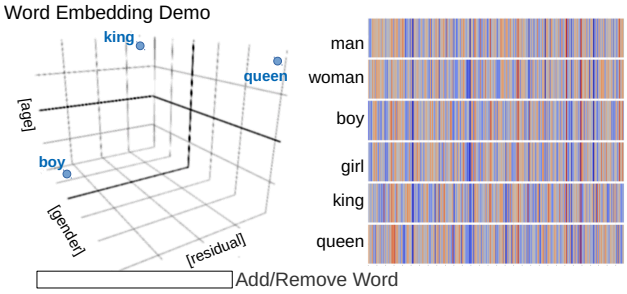

Figure 1: Schematic view of our interactive embedding visualization demo. On the left side is a zoomable and rotatable 3D scatter plot. On the right is a display of six user-selectable embedding vectors. For detailed views see Figures 2 and 3. The text box at the bottom allows entry of new words. Below that, expandable panels (not shown) reveal more advanced features, such as analogies.

NLP includes a variety of syntax, semantics, discourse, and speech tasks that can take advantage of word embeddings. Examples include text classification and categorization, named entity recognition, part of speech tagging, semantic parsing and question answering, paraphrase detection, language generation, multi-document summarization, and machine translation.

Transformers, a class of deep neural network architectures with built-in attention mechanisms, operate on text by taking a sequence of word embeddings as input and producing another sequence as output. They are used today for Neural Machine Translation (NMT), question answering, and language generation tasks. Cutting-edge question answering and language generation models such as BERT (Devlin et al. 2018) and GPT3 (Brown et al. 2020) are built using transformers.

We have developed an interactive tool for visualizing and exploring word embeddings that runs purely in the browser (Bandyopadhyay et al. 2021). See Figure 1.

Word2Vec Architecture

Early work on word embeddings was based on Latent Semantic Analysis, which calculated a co-occurrence matrix for words drawn from a corpus of text and then used princi-

pal components analysis to derive a modest-sized vector for each word.

Word2Vec is an adaptive algorithm that starts with random word representations and uses gradient descent to adjust those representations to improve its co-occurrence predictions (Mikolov et al. 2013a). There are two versions of the word2vec algorithm: CBOW (Continuous Bag of Words) and Skip-gram. We focus here on Skip-gram.

The Skip-gram algorithm creates two sets of word vector representations, one for the words themselves (called the embedding vectors), and one for words appearing in the context of other words (the context vectors). The size of these vectors is a parameter to the algorithm; it typically ranges from 100 to 300 elements. The context window size is also a parameter, with a value between 2 and 50 words. For a window of size 5, any given word in the text will have the two words before it and the two words after it as its context.

Initially both the embedding and context vectors are random. As the algorithm cycles through the words of the corpus, it trains on both positive and negative examples. Positive examples are words that have appeared at least once in the context window. Negative examples are words that have never appeared in the context of the current word. The algorithm takes the dot product of the subject word's embedding vector and the context (or non-context) word's context vector and runs the result through a sigmoidal nonlinearity, giving a value between 0 and 1. The desired output is 1 for context words and 0 for non-context words. The difference between the actual and desired output is an error signal that is used to adjust both the embedding vector and the context vector.

After some number of iterations, learning terminates and the embedding vectors are the result. The context vectors are discarded. Further details and an illustrated tutorial are provided in (Alammar 2019).

fastText (Bojanowski et al. 2017) is a refinement of the word2vec algorithm that decomposes words into an n-gram letter encoding instead of assigning each word a unique one-hot code. It produces slightly poorer performance than Skip-gram or CBOW on word analogy problems, but has the advantage of being able to derive embedding vectors for words not in the training set, based on their n-gram representations.

The initial version of our demo used 100-dimensional vectors trained from the text8 corpus using Skip-gram (Mahoney 2006a). This corpus is the first 100MB of clean text data obtained from the English Wikipedia text dump for the Large Text Compression Benchmark (Mahoney 2006b). "Clean text data" refers to human-readable text that is visible on a Wikipedia web page. Our current demo uses 300-dimensional pre-trained word vectors without subwords, generated from a fasttext.cc dataset containing a mix of Wikipedia text and news stories (Mikolov et al. 2018). Only words with purely alphabetical characters were used, and for words with multiple capitalizations (treated as different words in the dataset), the one with the highest frequency was used.

For performance reasons we limited the vocabulary to the 50,000 highest frequency entries, and precomputed the 10 closest words for each one. However, when performing vector arithmetic on words, the 10 nearest words to the result vector (e.g., "king" minus "man" plus "woman") must be computed on the spot.

Previous Work

We found 11 interactive online demos dealing with word embeddings. We list below the various activities these demos support. Table 1 summarizes the activities afforded by each demo.

- *Find nearest words.* Given a word, find the words closest to it in vector space, using either Euclidean distance or cosine similarity metrics.
- *Word similarity.* Given two words, estimate their similarity by comparing their vectors.
- *Analogies.* Complete the analogy $A{:}B :: C{:}?$ by finding the word D nearest to $B - A + C$.
- *Outlier detection.* Given a set of words, such as "dinner", "cereal", "breakfast", and "lunch", determine which one does not fit with the others by computing their distances in vector space.
- *Concept projection.* Define a semantic dimension by picking two contrasting words, such as "food" and "pet", and subtracting their vectors. Then project a collection of words onto that axis. May be done in either one or two dimensions.
- *Co-occurrence analysis.* Comparing a set of word vectors based on their co-occurrence statistics with related words. For example, two female words "aunt" and "sister" may have similar co-occurrence statistics over a set of words that includes "married", "love", "actress", "king", "son", "daughter", etc.. And two male words "uncle" and "brother" may also have similar co-occurrence statistics. But the co-occurrence statistics for "man" may not look like the other male words, and those for "woman" may not look like the other female words. This suggests that "uncle" minus "aunt" or "brother" minus "sister" might be a better proxy for gender than "man" minus "woman" (Heimerl and Gleicher 2018a).
- *1D visualizaion.* Projection of words onto an axis based on cosine similarity to a reference word.
- *2D or 3D scatter plots.* Individual words are plotted as points in a 2D or 3D space. This can be done using either an embedding algorithm such as tSNE, or by projecting the points along dimensions such as selected principal components.

Most of these demos use English vocabulary, but (Robo Report 2016) uses a Korean Wikipedia dataset, (Kutuzov et al. 2017) offers both English and Norwegian vectors, and (Kutuzov and Kuzmenko 2017) offers both English and Russian.

The (Duman et al. 2017) demo highlights gender bias in word embeddings by having the user choose a male/female contrast pair such as "he"/"she", and also a set of seed words. It then finds a collection of words related to the seeds and shows, via both a word cloud and a tabular list, which words are more strongly associated with the male contrast

Demo	Nearest	Similar	Analogy	Outlier	Project.	Co-occ.	1D	2D/3D
(Turku NLP Group 2015)	x	x	x					
(Karem 2020)								x
(Liu 2016)	x		x					
(KB Labs 2019)	x		x					
(Heimerl and Gleicher 2018b)					x	x	x	x
(Smilkov, Thorat, and Nicholson 2014)	x				x			x
(Rehurek 2014)	x		x	x				
(Robo Report 2016)	x	x						
(Duman et al. 2017)	x							
(Kutuzov et al. 2017)	x	x	x					x
(Kutuzov and Kuzmenko 2017)	x	x	x					x
Our demo	x		x		x			x

Table 1: Interactive word embedding demos and their features. Column headings are described in the text.

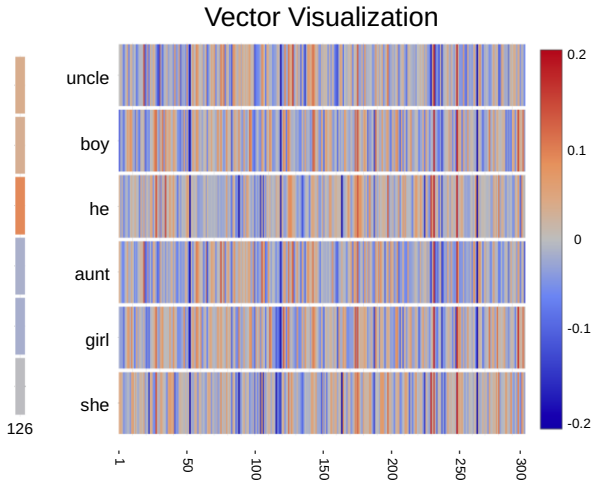

Figure 2: Embedding vectors for three male words ("uncle", "boy", "he") and three female words ("aunt", "girl", "she"). Component 126, shown magnified at left, is positive for the male words and negative for the female words.

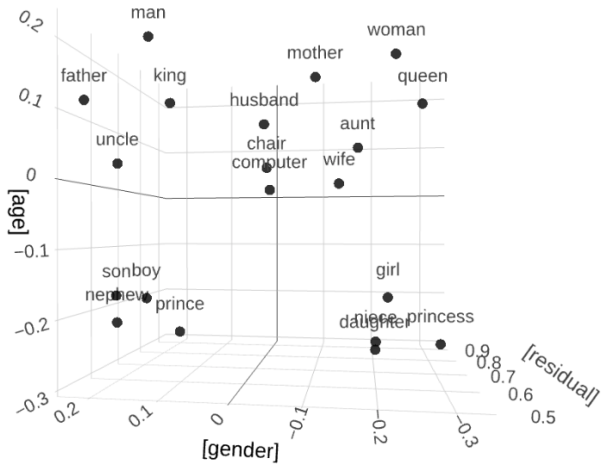

Figure 3: Words plotted in our 3D semantic space. Male words appear in the positive (left) half of the x-axis; female words in the negative (right) half. Adult words are in the positive (top) half of the y-axis; youth words in the negative (bottom) half. The third dimension is the "semantic residual", explained in the main text.

word, and which are more strongly associated with the female word.

Graphical Display of Embeddings

In some cases, semantic features in embedding vectors can be discovered by examining the vectors directly. Figure 2 shows the vectors for three male words and the corresponding female words. Element 126, shown magnified, is one of several that seem to correlate with gender. Semantic information is typically distributed across multiple vector components and is difficult to discern by looking at the vectors directly. In other words, the semantic dimensions we care about are not generally aligned with the coordinate axes.

Displaying the vectors directly is still useful for some purposes, but does not help the user appreciate the geometry of the semantic space that makes analogy by vector arithmetic possible. If each word is a point in the space, direct display of the vector for that word, as in Figure 2, is like writing out its coordinates. What we want to do instead is *graph* the point.

There are several approaches to generating graphical displays of word vectors. The simplest is to designate particular components out of the 100 to 300 available as the x, y and z values of the 3D graphical display. The problem, as we've seen above, is that individual components may not correlate strongly with the semantic features of interest. Thus, appropriate components may be difficult to find, and using single components will likely produce poor results.

Another approach is to perform Principal Components Analysis (PCA) on the vectors and then allow the user to select components to serve as the x, y, and z axes of the graph. The TensorFlow Embedding Projector demo (Smilkov, Thorat, and Nicholson 2014) offers this option. The problem

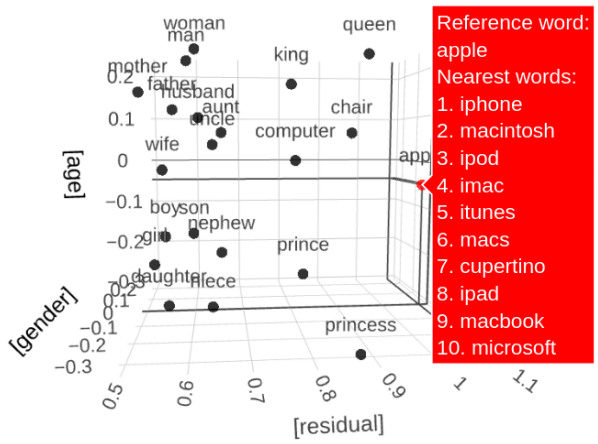

Figure 4: Rotated view of the scatter plot, showing distance along the residual dimension. Contrast pair words, which define the residual dimension, have small residuals, except for royal words which have moderate residuals. "Computer", "chair", and "apple", which are semantically unrelated to the contrast words, have high residuals. Hovering over the word "apple" pops up a list of the 10 closest words, which include "iphone", "macintosh", "cupertino", and "microsoft".

with this is that while PCA will likely find dimensions that encode meaningful differences between words, there is no guarantee that those dimensions will correlate strongly with the particular semantic features we're interested in.

A third approach to displaying semantic space is to use an algorithm such as t-distributed Stochastic Neighbour Embedding (tSNE) to embed the feature vectors into either a 2D or 3D space. The TensorFlow Embedding Projector also offers this option. Words get clustered together with this approach, but it does not effectively illustrate the high dimensionality of semantic feature space.

A fourth approach is to allow users to define their own semantic dimension by specifying a pair of contrasting words and subtracting their vectors. Words can then be projected onto the axis defined by this difference vector. The EmbVis system's Concept Projection demo (Heimerl and Gleicher 2018a) works this way, projecting a collection of words onto a 1D line. It actually does this twice to allow comparisons across corpora, e.g., seeing how animal words fall on the food-pet dimension for embeddings trained on Wikipedia data vs. those trained on EEBO-TCP (Early English Books Online Text Creation Partnership).

Following a similar approach, TensorFlow Embedding Projector (Smilkov, Thorat, and Nicholson 2014) allows users to specify two pairs of contrasting words, and then projects a selected collection of words into a 2D space defined by those two contrasts.

Our Approach

Our approach to displaying embedding vectors in 3D also uses two dimensions to represent selected semantic features such as gender and age. But it introduces a third "residual" dimension that encodes distance along the remaining axes of semantic space. For example, given the word "king", we measure its position along the gender axis by taking the dot product of its embedding vector with a "gender" unit vector. We perform a similar calculation to measure position along the age axis. The procedure for measuring position along the "semantic residual" axis will be described below. Given these three coordinate values, "king" can be plotted in 3D semantic space.

To create the gender and age unit vectors, we use multiple pairs of contrasting words, subtracting one from the other and averaging the results. For example, "man" minus "woman" should point in roughly the same direction as "king" minus "queen". Similarly, an age unit vector can be derived from pairs such as "man" minus "boy", "woman" minus "girl", and "king" minus "prince". Figure 7 shows the complete list of word pairs used to define each semantic dimension. We average the difference vectors and normalize the result to derive a unit vector along any desired semantic dimension.

To calculate the direction of the residual unit vector, we take each word used to define age or gender and subtract off its age and gender components, i.e., its projections along the age and gender axes. All of the contrasting words describe (or can be used to describe) people, so what's left after subtracting out their age and gender projections and averaging the results should be a genderless, ageless "person" vector $\vec{P}$. Other kinship terms such as "grandfather" should have a strong positive dot product with $\vec{P}$, while unrelated words such as "computer" should not. Thus, we calculate the residual coordinate r of a word $\vec{W}$ as $r = 2 \times (1 - \vec{W} \cdot \vec{P})$ so that a larger value corresponds to greater semantic distance from $\vec{P}$. The factor of 2 magnifies the residual for plotting, to visually emphasize that it's collapsing many semantic dimensions into one.

Using the Demo to Explore Word Embeddings

One of our core design principles is that a demo should display something interesting from the moment it's loaded, without requiring the user to take any action. So we have pre-defined the gender/age semantic space and added the vectors for a select group of words to the 3D scatter plot on the left. To the right, we display the 300-element vectors for six selected words: "man", "woman", "boy", "girl", "king", and "queen". Figure 1 shows the initial display when the demo is loaded.

Continuing our theme of providing a low threshold for informative interactions, simply moving the mouse pointer over a point in the scatter plot produces a pop-up window displaying that word and its 10 closest neighbors (Figure 4), measured by 300-dimensional Euclidean distance. Exploring the neighbors illustrates how dependent the vectors are on the choice of training corpus. For the vectors we're using, derived from Wikipedia articles and news stories, the closest neighbors of "apple" include "iphone", "mackintosh", "ipod", "cupertino", and "microsoft", not fruits.

Clicking and dragging on whitespace in the 3D plot ro-

tates the view, while the scroll wheel can be used to zoom in or out. This gives the user a better sense of words as points in a "space", with relative motion of points providing a helpful cue for depth. The graphics are produced using Plotly (Plotly Technologies Inc. 2015).

Displaying the word vectors as heat maps in the right half of the display (Figure 1) facilitates visual comparisons, but we also want to be transparent about the values. Moving the mouse pointer over an element of the word vector display produces a pop-up showing the word, the position of the element in the vector (0 to 299), and the precise value of that element, which typically ranges from -0.2 to +0.2.

Clicking on a word causes it to become "active", and its dot turns red. Only one word can be active at a time. When a word is active, clicking on a row of the vector display in the right half of the window causes that word's vector to be copied into that row. In this way, the six words initially shown in the vector display can be replaced with words of the user's choice. This makes it easy to visually compare vectors of related or contrasting words, as we've done in Figure 2.

New words can be added to the scatter plot by typing them into a text box at the bottom of the window. When a word is entered, it is made the active word; the red dot helps the user to locate it in the display. A previously-added word can also be entered in the text box to make it active again if the user needs help finding it.

The ability to add selected words to the plot permits a variety of informative exercises:

- Given a new word that could be part of a contrasting pair, such as "grandfather", try to predict where it will appear in the 3D space. Is it further along the positive age dimension than "father"?
- After plotting a word that could be part of a contrasting pair, try to predict where its contrasting partner will appear. For example, where does "grandmother" lie relative to "grandfather"? Where does "grandson" lie?
- How do related words cluster together, e.g., "apple", "orange", "banana", "peach", "plum"?
- How do alternate forms of a word cluster together, e.g., "peach" vs. "peaches", or "eat", "eats", and "ate"?

Analogy by Vector Arithmetic

To simplify the visual presentation and avoid intimidating inexperienced users, the advanced features of our demo are hidden in collapsible panels. The first of these is the Vector Arithmetic Analogy panel. Expanding this panel reveals text boxes with the now classic example "king" minus "man" plus "woman". If the user types the suggested text into the boxes, the calculation is performed and the results displayed. In the 3D plot, the three source words are plotted as blue points, and the result of the arithmetic operation is plotted as a pink point. This result will not exactly match any of our vocabulary vectors, so, following the procedure in (Levy, Goldberg, and Dagan 2015) we find the closest word vector to it that is not one of the source vectors, and plot that word in green. For this example, the result is "queen". We draw arrows from "man" to "king" and from "woman" to "king - man + woman", as shown in Figure 5. Drawing the arrows clearly shows the analogy, and the ability to interactively zoom and rotate the display makes it easy to confirm that the two arrows are parallel.

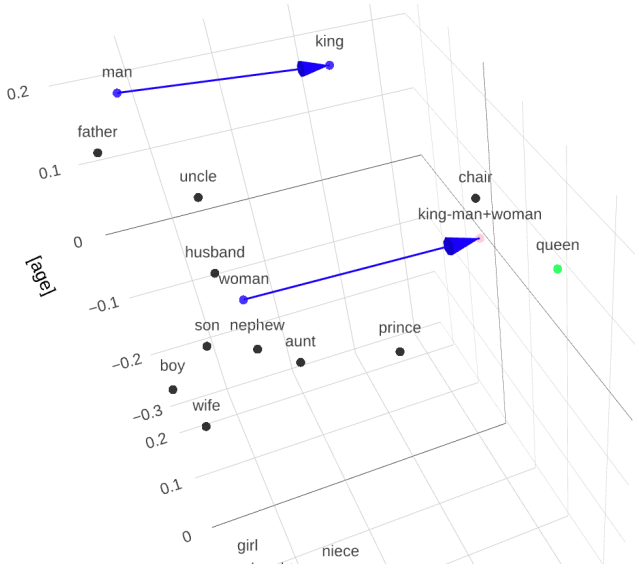

Figure 5: Analogy by vector arithmetic: "man" is to "king" as "woman" is to "king − man + woman" = "queen".

In the right half of the display (Figure 6), all the vectors for the computation are shown, i.e., "king", "man", "king - man", "woman", "king - man + woman", and the closest match, "queen".

Many different types of analogies can be demonstrated successfully using this procedure. For example, "Paris is to France as London is to England". Several thousand analogies are discussed in (Mikolov et al. 2013a; Mikolov, Yih, and Zweig 2013). Other examples include "finger is to hand as toe is to foot", "see is to saw as eat is to ate", and "cow is to beef as sheep is to lamb". ("Mutton" would be more correct, and that is the second closest answer found.). But there are failures as well, e.g., "girl is to princess as woman is to X" produces "prince" as the closest match, although the next two choices, "empress" and "queen", would be satisfactory. In the 3D plot, "queen" appears much closer to the result vector than "prince" does, but this is misleading: while "queen" is closer along the age and gender dimensions, "prince" may be closer along other semantic dimensions, all of which are being compressed into the residual. Note that the choice of semantic dimensions affects only the graphical display, not the analogy calculation itself.

Geometry analogies such as "square is to cube as triangle is to pyramid" do not appear to work using the embeddings dataset we've selected.

Defining Semantic Dimensions

A second hidden panel reveals the list of semantic dimensions. The two initially selected are *[age]* and *[gender]*, but we also provide *[royalty]*, *[number]*, *[part-of]*, *[tense]*, *[capital]*, and two blank dimensions. Expanding a dimen-

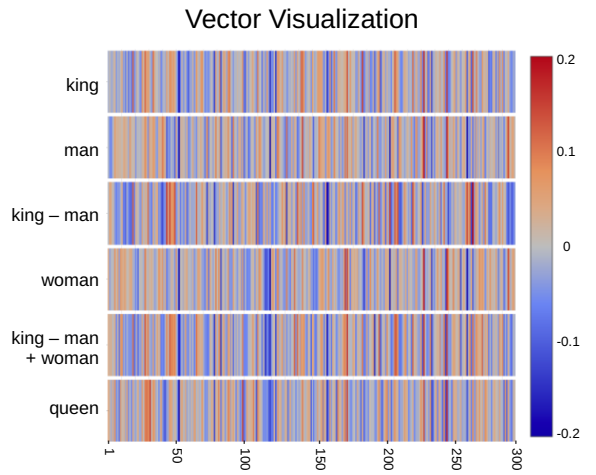

Figure 6: Steps in the vector analogy computation. The six rows of the display show the vectors for (1) "king", (2) "man", (3) "king" minus "man", (4) "woman", (5) "king" minus "man" plus "woman", and (6) "queen".

sion reveals its list of opposed words. The word pairs for several of our built-in dimensions are shown in Figure 7. Users are free to edit these lists to experiment with different definitions for a dimension. Note that for *[royalty]* we avoided terms such as "count" and "duke" whose multiple meanings might distort the result.

Building in these dimensions lowers the threshold for experimentation, which is important for younger students. It allows them to explore both the generality of a mapping (e.g., trying different part-whole analogies) and the basis of the mapping (e.g., trying different sets of opposing word pairs as the basis of *[part-of]*).

Figure 8 shows the same vector analogy as before, but now plotted in the gender×royalty semantic space. This change also affects the residual dimension.

Discussion

Word embeddings are important because they are used in many state of the art natural language processing systems. The ability of embeddings to capture some semantic information allows a reasoner to make analogies or measure semantic similarity using simple vector arithmetic. However, semantic accuracy is not guaranteed, and in fact is impossible to achieve because of polysemy and homographs. Words such as "lead" are both nouns and verbs, and even the noun has multiple unrelated meanings. Natural language processing systems trained on huge corpuses can disambiguate these meanings based on context, but the embedding vector itself cannot. The unique distributional characteristics of a word like "lead" will distinguish it from other words in the vector space, but at the cost of muddying its semantic content.

Our demo helps students experience both the strengths and limitations of word embeddings by doing their own experiments. The interface is designed to facilitate exploration

Gender Dimension	
man	woman
king	queen
prince	princess
husband	wife
father	mother
son	daughter
uncle	aunt
nephew	niece
boy	girl
male	female

Age Dimension	
man	boy
woman	girl
king	prince
queen	princess
father	son
mother	daughter
uncle	nephew
aunt	niece

Royalty Dimension	
man	king
woman	queen
boy	prince
girl	princess
woman	duchess
woman	countess
woman	baroness

Figure 7: Some semantic dimensions defined by pairs of opposed words. Gender and age are the default dimensions when the demo is started. All dimensions are user-modifiable via editable text boxes.

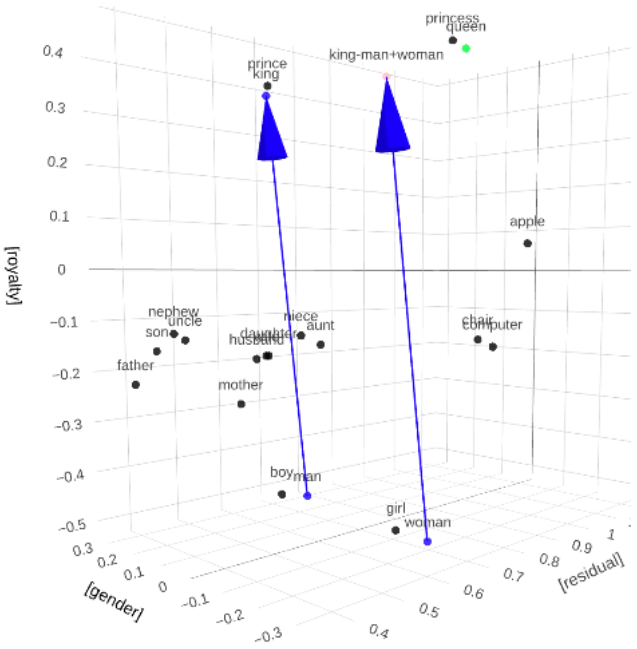

Figure 8: Replacing the *[age]* dimension with a *[royalty]* dimension dramatically changes the display, but does not affect the actual analogy computation.

and not overwhelm the user with options and parameters.

Initial User Testing

We tested the demo on two high school students with programming experience and one undergraduate computer science major. None had heard of word embeddings before. The high school students were sent an article on word embeddings (Alammar 2019) and then asked to play with the demo on their own, with no further instruction. They were interviewed about their experience afterwards. The undergraduate was not shown the article but was verbally guided through the demo.

All students found the 3D scatter plot to be intuitive. They were able to move about the space using rotate and zoom, and they discovered on their own that hovering over a point produced the list of closest words. None of them discovered on their own that a word could be copied into one of the six vector display positions by clicking on the point in the 3D plot and then clicking on a row of the vector display. This is not surprising, as the user interface provided no affordances for this. We have since modified the demo to provide visual prompting for this operation, and will also provide written operating instructions.

All the students grasped the concept of analogy by vector arithmetic and were appreciative of the arrows drawn in the scatter plot when they ran the "king minus man plus woman" example. But all three expressed difficulty formulating analogies of their own using the $A-B+C = D$ arithmetic convention provided by the demo. They were much more comfortable with an "*A* is to *B* as *C* is to *D*" formulation, which corresponds to $B - A + C = D$. Swapping the A and B terms and correctly identifying the C term were problematic when translating between analogy and equation forms. We have modified the demo to allow entry in either notation and translate automatically between the two.

The text boxes for user-defined semantic dimensions were not interpretable by the students despite the fact that they came pre-populated with the gender and age word-pairs. While some improvements to the visual layout were made afterwards, it's clear that this portion of the demo will require explanatory text.

Additional Experiments

Earlier we described some experiments where students investigate how related words cluster together in semantic feature space. Here we list some more advanced topics for students to explore.

Polysemous words such as "man" work less well in analogies. Students should be prompted to think about why this is the case. (It's because their representation is a blend of the different contexts where they are used.) Thus, "man is to woman as stallion is to [mare]" fails in our demo. Students can be asked to investigate other words for referencing gender. "Uncle is to aunt" works better than "man is to woman" for the stallion analogy.

While "doctor" and "nurse" are gender-neutral terms, some corpuses reflect cultural biases that allow for analogies such as "uncle is to aunt as doctor is to nurse", or even "as strong is to weak" or "as hairy is to hairless", all of which our demo produces using the current fastText embedding. Looking for examples like this is a good way to get students to think about language and culture.

Semantris

Semantris (Google Research 2018) is an online word association game. The name is a play on "Tetris" as the game involves eliminating blocks. In Semantris some blocks contain a word, and blocks are removed by typing a list of associated words, e.g., for a "piano" block one might enter "music, keyboard, pedals, fingering". Word association is measured using embeddings that were produced by Google research projects on language understanding (Cer et al. 2018).

Semantris has proven to be a good way to introduce both students and teachers to the topic of word embeddings. After playing the game several times, they can be prompted to think about how the game knows that a word like "piano" is associated with "music" and "keyboard". This leads to a discussion of semantic spaces, which our demo then illustrates.

Alignment with the AI4K12 Guidelines

This demo aligns with several of the AI education guidelines presented by AI4K12.org (AI4K12.org 2021). Within Big Idea 2, Representation & Reasoning, section A focuses on representations, and the 2-A-iv guidelines concern feature vector representations. The 2-A-iv.6-8 guideline explicitly references word embeddings, while 2-A-iv.9-12 references transformer networks, which utilize word embeddings.

Guideline 2-C-i examines types of reasoning problems. Analogy problems, although not explicitly considered in the guidelines, are a type of prediction problem. Solving analogies by vector arithmetic is a specific reasoning algorithm (guideline 2-C-ii) that can be employed when concepts are represented as feature vectors.

Big Idea 3, Learning, is relevant because word embeddings are constructed by machine learning algorithms. Guideline 3-C-iii examines how biases in datasets affect learning. This can be illustrated in the demo by looking at examples like "apple", where the closest words are not fruits, but computer technology words.

Future Work

We would like to offer alternative corpuses. Wikipedia articles, current news stories, and classical literature all have different word co-occurrence statistics which will lead to different embeddings. Also, newer algorithms such as GloVe (Pennington, Socher, and Manning 2014) and ELMo (Peters et al. 2018) may produce qualitatively different embeddings in terms of word similarity and analogy.

Another extension we are considering would rotate the 300-dimensional feature space so that the two user-defined semantic dimensions were aligned with the first two coordinate axes. This would make the difference between male and female words, or young and old words, visually salient in the feature vector display. Seeing what this transformation accomplished would give students a deeper understanding of mathematical operations on feature vector representations.

Acknowledgments

This work was supported by National Science Foundation award DRL-2049029.

References

AI4K12.org. 2021. AI4K12 Grade Band Progression Charts. https://ai4k12.org/gradeband-progression-charts. Accessed: 2021-09-08.

Alammar, J. 2019. The Illustrated Word2vec. https://jalammar.github.io/illustrated-word2vec/. Accessed: 2021-09-08.

Bandyopadhyay, S.; Xu, J.; Pawar, N.; and Touretzky, D. 2021. Word Embedding Demo. https://www.cs.cmu.edu/~dst/WordEmbeddingDemo. Accessed: 2021-12-10.

Bojanowski, P.; Grave, E.; Joulin, A.; and Mikolov, T. 2017. Enriching Word Vectors with Subword Information. arXiv:1607.04606.

Brown, T. B.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. 2020. Language models are few-shot learners. arXiv:2005.14165.

Cer, D.; Yang, Y.; yi Kong, S.; Hua, N.; Limtiaco, N.; John, R. S.; Constant, N.; Guajardo-Cespedes, M.; Yuan, S.; Tar, C.; Sung, Y.-H.; Strope, B.; and Kurzweil, R. 2018. Universal Sentence Encoder. arXiv:1803.11175.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2018. BERT: Pre-training of deep bidirectional transformers for language understanding. arXiv:1810.04805.

Duman, S.; Kalai, A. T.; Leiserson, M.; Mackey, L.; and Sursesh, H. 2017. Gender bias. http://wordbias.umiacs.umd.edu/. Accessed: 2021-09-08.

Google Research. 2018. Semantris. https://research.google.com/semantris. Accessed: 2021-09-08.

Heimerl, F.; and Gleicher, M. 2018a. Interactive analysis of word vector embeddings. *Computer Graphics Forum*, 37(3): 253–265.

Heimerl, F.; and Gleicher, M. 2018b. Interactive Analysis of Word Vector Embeddings. https://graphics.cs.wisc.edu/Vis/EmbVis/. Accessed: 2021-09-08.

Karem, R. 2020. Word2Vec Demo. https://remykarem.github.io/word2vec-demo/. Accessed: 2021-09-08.

KB Labs. 2019. Word2Vec-DIGHUMLAB. https://dighumlab.org/word2vec/. Accessed: 2021-09-08.

Kutuzov, A.; Fares, M.; Oepen, S.; and Velldal, E. 2017. Word vectors, reuse, and replicability: Towards a community repository of large-text resources. In *Proceedings of the 58th Conference on Simulation and Modelling*, 271–276. Linköping University Electronic Press.

Kutuzov, A.; and Kuzmenko, E. 2017. *WebVectors: A Toolkit for Building Web Interfaces for Vector Semantic Models*, 155–161. Cham: Springer International Publishing. ISBN 978-3-319-52920-2.

Levy, O.; Goldberg, Y.; and Dagan, I. 2015. Improving Distributional Similarity with Lessons Learned from Word Embeddings. *Transactions of the Association for Computational Linguistics*, 3: 211–225.

Liu, A. 2016. Word2Vec JS Demo. https://turbomaze.github.io/word2vecjson/. Accessed: 2021-09-08.

Mahoney, M. 2006a. About the Test Data. http://mattmahoney.net/dc/textdata.html. Accessed: 2021-09-08.

Mahoney, M. 2006b. Large Text Compression Benchmark. http://mattmahoney.net/dc/text.html. Accessed: 2021-09-08.

Mikolov, T.; Chen, K.; Corrado, G.; and Dean, J. 2013a. Efficient estimation of word representations in vector space. arXiv:1301.3781.

Mikolov, T.; Grave, E.; Bojanowski, P.; Puhrsch, C.; and Joulin, A. 2018. Advances in Pre-Training Distributed Word Representations. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*.

Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G. S.; and Dean, J. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26*, 3111–3119.

Mikolov, T.; Yih, W.; and Zweig, G. 2013. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics*, 746–751.

Pennington, J.; Socher, R.; and Manning, C. D. 2014. GloVe: Global Vectors for Word Representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, 1532–1543.

Peters, M. E.; Neumann, M.; Iyyer, M.; Gardner, M.; Clark, C.; Lee, K.; and Zettlemoyer, L. 2018. Deep contextualized word representations. In *Proc. of NAACL*.

Plotly Technologies Inc. 2015. Collaborative data science. https://plot.ly. Accessed: 2021-09-08.

Rehurek, R. 2014. Word2vec Tutorial by RARE Technologies. https://rare-technologies.com/word2vec-tutorial/. Accessed: 2021-09-08.

Robo Report. 2016. Korean Word2Vec & Doc2Vec Demo. http://stockprediction.co.kr/word2vec/. Accessed: 2021-09-08.

Smilkov, D.; Thorat, N.; and Nicholson, C. 2014. Embedding Projector. https://projector.tensorflow.org/. Accessed: 2021-09-08.

Turku NLP Group. 2015. Word embedding demo. http://bionlp-www.utu.fi/wv_demo/. Accessed: 2021-09-08.