Polyglot: 3D Language Embedding Visualization - Media-making Statement

Hongwei (Henry) Zhou

Overview

Polyglot is a web application for visualizing language embeddings in a 3D space. Language embeddings are typically high-dimensional vector representations of the syntactic and semantic content of words. This application allows examination of a particular word embedding data, reduced to 3D using UMAP. In addition to 3D navigation of the scatter plot space, the application also uses colors to enable two ways of exploring the dataset: (1) coloring based on the result of Monte-Carlo Physarum Machine (MCPM) and (2) coloring based on each word's part-of-speech tag.

This is a visualization tool developed during my second year as a PhD student at UCSC. It eventually became a part of my master thesis. In this media-making statement, I will divide the development history roughly into three phases: the beginning phase, the linguistic exploration phase, and the structural comparison phase. Each phase was propelled by different needs that arose and was defined by the features added in response to those needs. The beginning was propelled by our (Oskar Elek's and my) desire to have an interactive visualization tool for the Slime Mold in the first place. The linguistic exploration phase is characterized by features added based on the feedback by a linguistic professor Pranov Anand. The structural exploration phase was motivated by Adam Smith's desire to examine structural invariance in dimensionality reduction algorithms when I was putting together my master thesis.

Word Embeddings

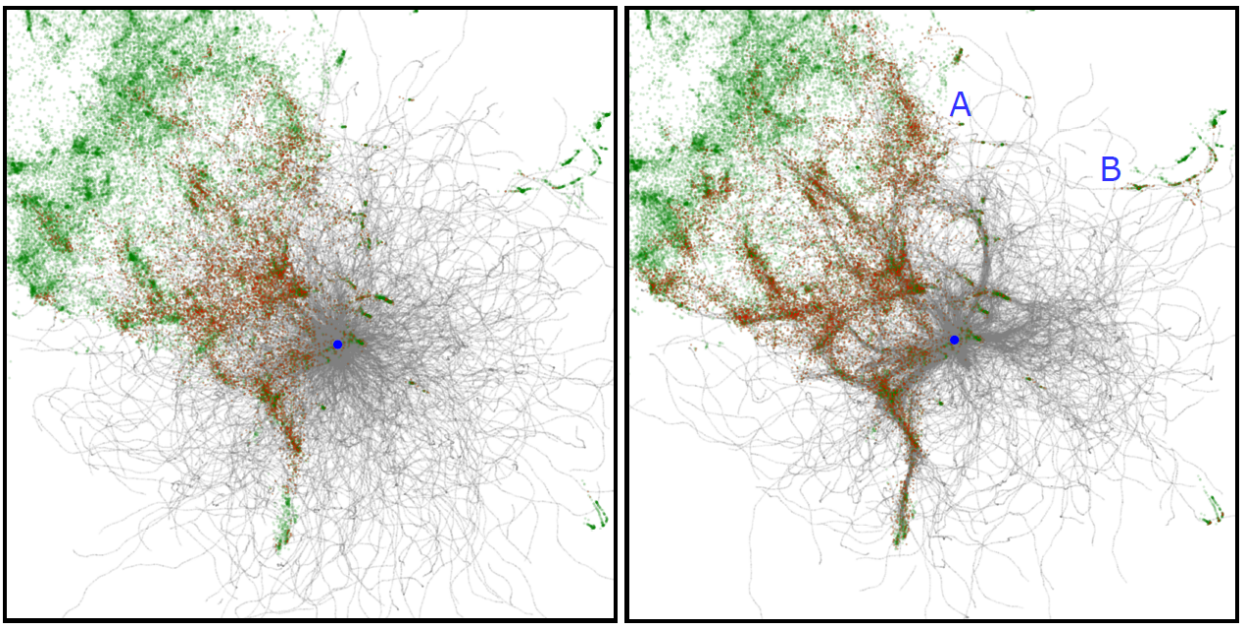
Word embeddings, such as Word2Vec, GloVe, and ELMo, are vector/points generated by an algorithm. The key computational idea is to transform topological information contained in a relational graph to geometric information encoded in a D-dimensional vector (`embedding') space. Embeddings have a number of interesting algebraic properties: most importantly, the contextual similarity of the embedded tokens is transformed into geometric proximity in the embedding [6, 7]. Because they explicitly consider the token's context [8, 9], it has been shown that embeddings contain information that can be processed to extract a range of useful properties: clustering by token usage [10, 11] as well as different kinds of syntactic information [10, 12, 13]. Thus, there is the promise that this kind of method could provide high-dimensional representations that encode a large manner of relations implicitly without having to hand-code them in advance.

But it's impossible to visualize embedding in its original format because of high dimensionality. Dimensional reduction techniques are applied so that the points can be represented in either 2D or 3D space. But reducing its dimensionality also means that much encoded information is lost. Visualizing embedding typically means getting a brief impression of its structure by proximity of different clusters. It can only remain brief impressions because of the loss of information, which means that information retrieval is not salient in lower-dimensional representations.

Applying Slime Mold to Word Embedding

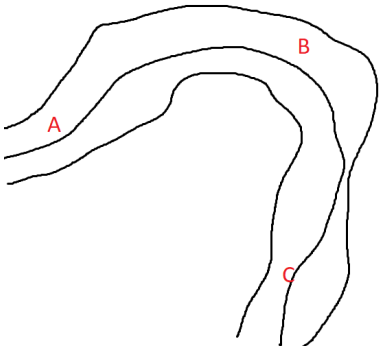
MCPM, aka. Slime Mold, is an agent-based model inspired by the self-organizing characteristics of slime mold, initially studied by Jeff Jones [1]. It was then modified by Burchett et al. with an additional Monte Carlo decision-making process, and was shown to be empirically accurate in predicting the pattern of the cosmic web of the universe [2, 3], where it has successfully recovered the theoretically predicted lamentary patterns over sparse galaxy data. MCPM can be understood as bio-inspired modeling of optimal transport networks. The mathematics of optimal transport [4, 5] is based on the principle of least effort, which applies to phenomena ranging from particle and light transport to the behavior of living beings.

An intuitive way to think about MCPM is that it discovers structures by spawning a swarm of agents. The agents navigate by following the trace - a density field in a 3D grid, implemented with a 3D array. The value within each cell of the trace determines how much the agent is attracted to it. Thus, the agent's travel trajectory is steered by the density value in the trace. The density is determined by two things: (1) When a "food", or a "galaxy", or "word embedding" is placed down, it contributes significantly to the density value and (2) Each agent deposits a small amount of density value into the trace as well. A standard use of MCPM is to spawn a swarm of agents, and run the simulation until the agents' travel paths eventually stabilize, forming discernable highways and roads. After stopping the simulation, the trace is then extracted as the result. The below figure demonstrates the impact of trace navigation. The left shows the exploration path for unguided agents, while the right shows the path while agents are guided by the trace. The agents are spawned at the same starting position (blue dot) and their trajectories are marked in gray. They are set out to discover the green data points (food/galaxy/word embedding), which are marked in red when discovered.



One can see the impact of the trace guiding (right), in comparison to unguided, purely random search (left). With trace guiding, most agents follow a few distinct paths to discover the surrounding token clusters. Without guiding, the random-walk process ends up being equivalent to the nearest neighbor search: the likelihood of a token being discovered decreases as a square of distance from the origin, as the agents become more spread-out. The two marked regions A and B in the right subfigure illustrate this contrast: from the random walk density we see that region A is more thoroughly explored than B in spite of both having a similar Euclidean distance from the source. This translates to A being closer within the paradigm of optimal transport.

What originally motivated us to apply MCPM to explore low-dimensional word embedding data is precisely this structure identification capability through connectivity of the point data. What is notable here is how connectivity differentiates itself from euclidean-distance-based structural identification. We'll explore the low-dimensional results generated by a dimensionality reduction algorithm called UMAP [15]. It organizes higher dimensional points into filaments and clusters based on their proximity in the higher dimension, and then tries to fit those filaments and clusters in lower dimension. Looking at the illustration below, point A has roughly the same distance to B and to C. However, B is considered closer to A if we consider the connectivity of the data, which makes MCPM a salient tool for exploring structures in lower-dimensional data.

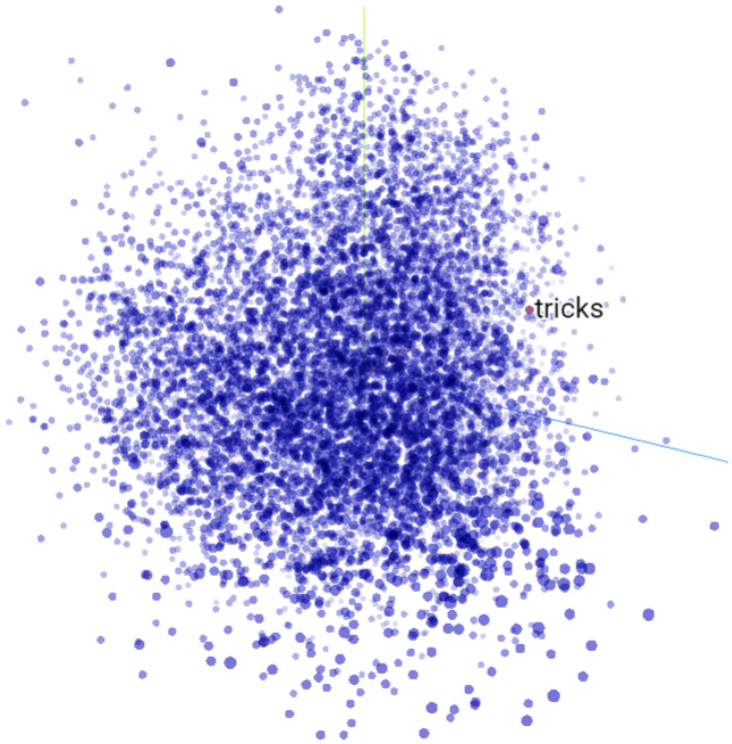


This project is slightly different from MCPM because the trace is fixed. We'll call the agent swarm MCPM probe agents instead because it does not deposit into the trace. The pipeline of combining MCPM and word embedding is the following: (1) MCPM simulation over word embedding data, extract trace once it stabilizes, (2) Use MCPM probe agents by spawning agents *on top of a single token* (we'll call this token *anchor point*). The agents will follow the extracted, fixed trace. When another word embedding is close enough, a counter for that word embedding is incremented. (3) We extract the counter for each word embedding. The higher the counter, the more similar to the chosen anchor point that word embedding is.

Initial Inspiration and Implementation

The visualization figure in the above section was created by running a Python script. The downside is rather obvious: we need to decide where the agent spawns and the view window by

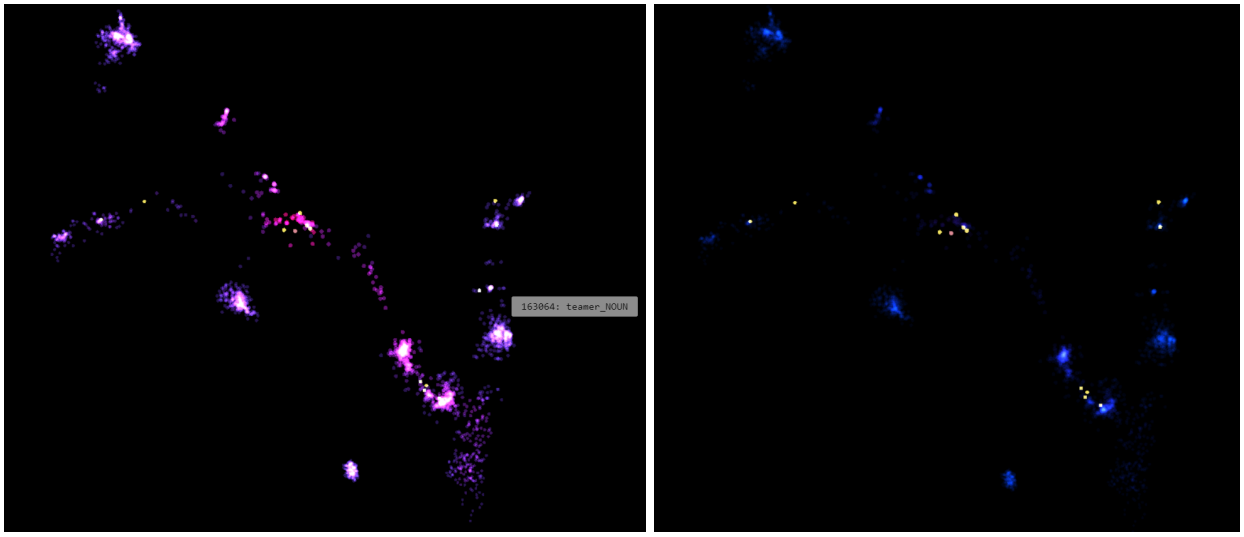
modifying the script, which does not provide us with immediate visual feedback. In addition, this view cannot tell us which word tokens are discovered (red). It became obvious that a new visualization tool is needed for us to explore the MCPM simulation result further. One major inspiration is the Tensorflow Embedding Projector. The screenshot is provided below,



It is a rather intuitive tool to explore word embedding in 3D space: mouse control to navigate within the 3D space, including camera rotation, translation and scaling. In addition, the mouse pointer hovering over a point reveals the content of that token.

We decided on using Three.js to implement our own version. Because of the rendering and camera control library already provided, the initial version works very similarly to Tensorflow Embedding Project.

We're interested to use this tool to examine the MCPM probe result. As mentioned in the pipeline above, the probe result is relative to the anchor point we choose (the point to spawn probe agents on). So we need to solve two problems: (1) How do we represent probe results aka. the counters? (2) How do we switch between different anchor point results? The result screenshot is shown below on the left figure.



For the probe result problem, we decided to map the counter to a color spectrum - from bright pink to dark blue, with a slider that controls the interpolation curve. For the switching between anchor points problem, I decided to color the anchor points yellow. When the user double clicks on them, the system will load the probe result for that anchor point. Additionally, I added a feature to help users select the anchor points, because the points can be rather cluttered. When users hold down the left Shift key, all non-anchor points are dimmed and the mouse pointer can only select the anchor points. A screenshot demonstrating this feature is shown on the above right figure.

From Local to Global

At the beginning, the tool only rendered points close to the anchor point. The original thinking was that since there are half of a million points, it would waste space to store the counters of all the word embedding for every anchor point. So the data format is the following:

[id, x, y, z, word_string, counter]

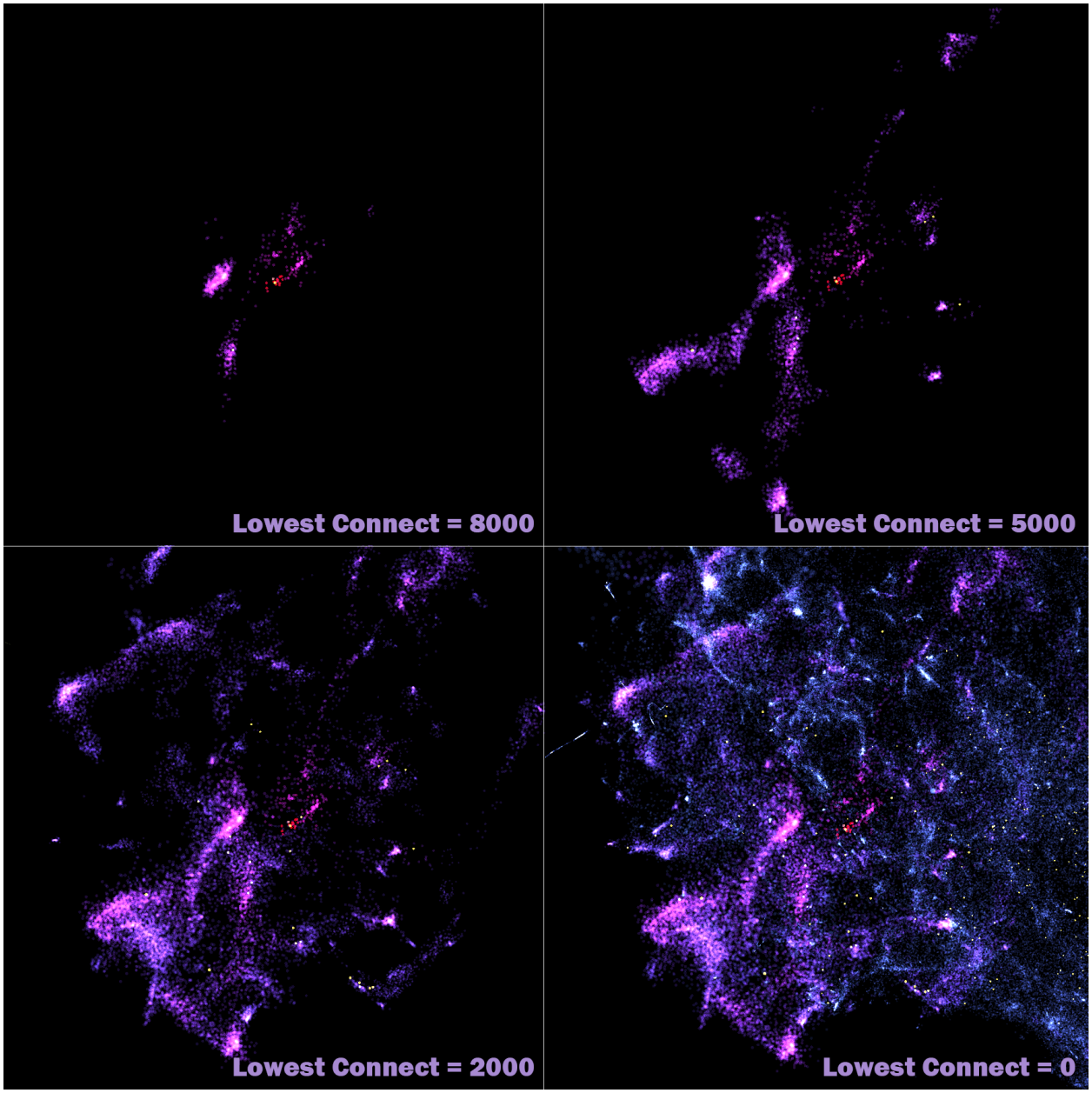
Each probe result for an anchor point is its own file, with around 5000 lines of data in the above format. As one can see, it stores all the information needed to render that particular point.

The downside is that the user will always get a local view of the data, never the overall shape of the data itself. In addition, the user cannot see all the anchor points they can jump to. Their options are limited to the anchor points discovered within the local probe result. This might make an interesting game idea, but not for data visualization.

Eventually, I decided to restructure the entire data loading pipeline. The position data and word string are stored in a single file. The tool loads all of it on startup. Each anchor point probe still results in its own file. But the data format becomes the following:

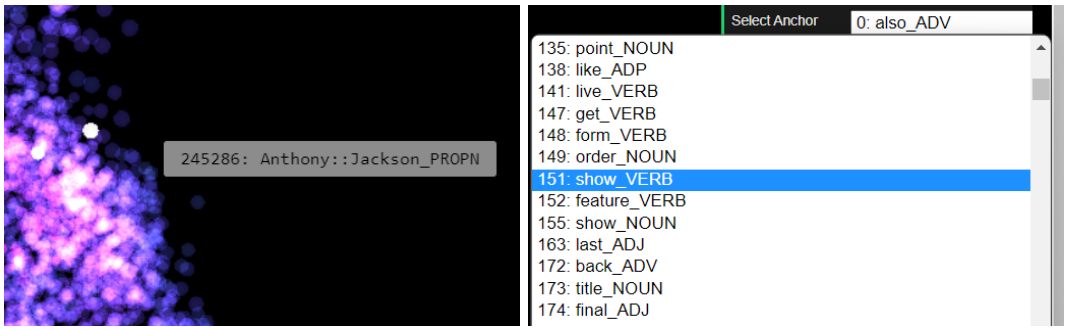
[id, counter]

I use the id, which is also the index for the word embedding, to extract the positional and word string data. At the end, the tool can show all of the dataset, whether the points are discovered during probing or not. I added a *Lowest Connect* slider for the user to control how much they can see. When its value is 0, the tool shows all the points. A figure demonstrating the difference is shown below.

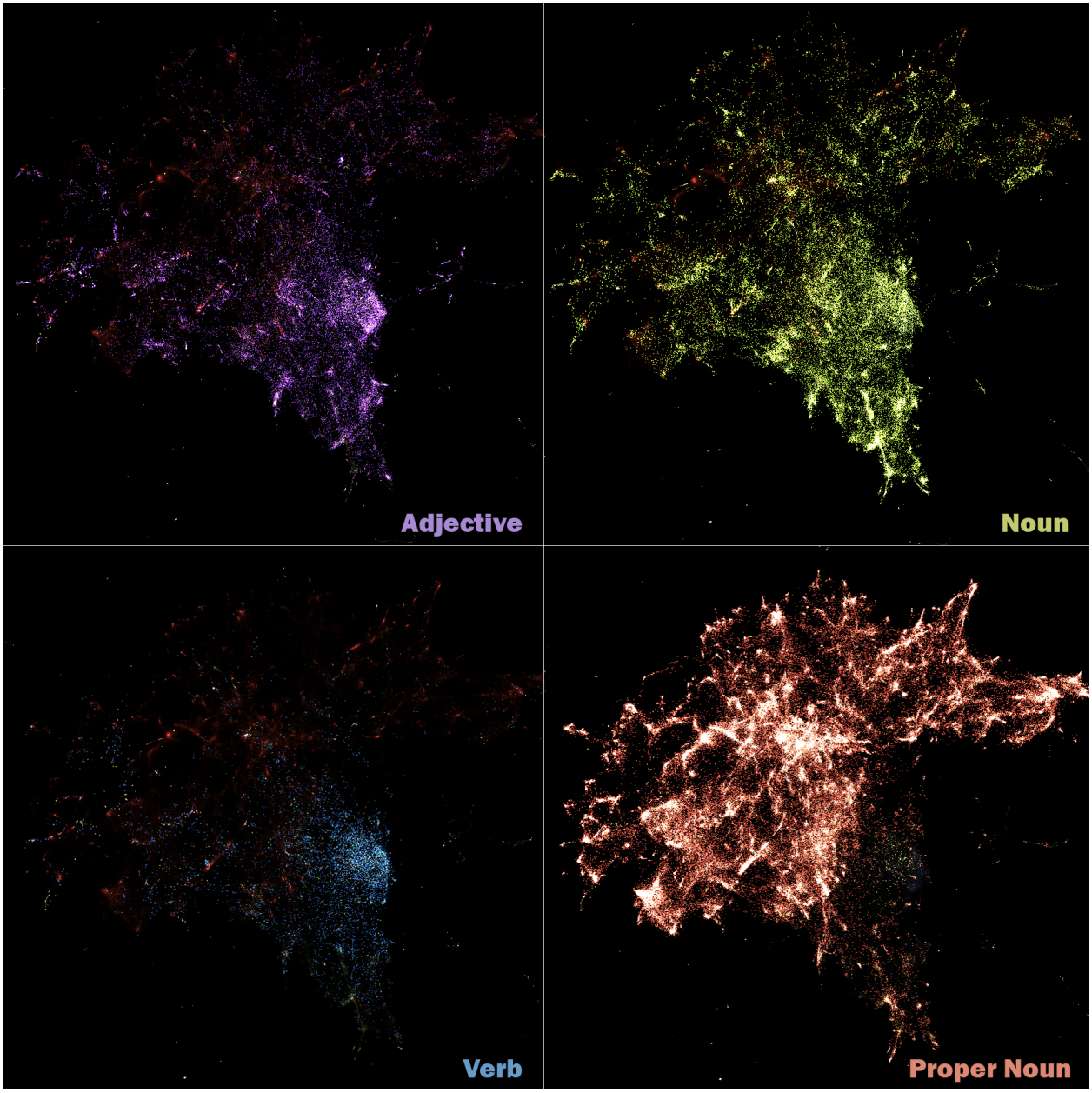


Pranov Anand: “I wish I could go back to where I was, also it would be nice to show the distribution of different word categories.”
The above is not an actual quote, but does represent the feedback we received. Pranov Anand is a linguistic professor whom we consulted several times on this project. After finishing a usable version of the tool, we shared it with him to see if a linguist would find it interesting or useful. Having significantly less 3D navigation literacy, Pranov struggled quite a bit trying to make sense of the camera control, and to grasp the shape of the 3D scatter plot. Rotating the camera seemed like a confusing morphing of the screen that he found overwhelming. This was something that we unfortunately did not consider and could not address properly, because 3D navigation is such a central part of our tool.

Even so, Pranov still spent quite a bit of time exploring the dataset. He found some interesting clusters in the dataset that allowed him to reason about how the algorithm and the dataset works. But because the dataset is so large, it was very difficult to reproduce the discovery by relocating where the clusters are. The solution we came down to was: (1) When hovering over points, the id of the point will also be displayed, (2) Have a dropdown menu so the user can quickly switch between anchor points, ordered by the id number. The intended solution is that the user can identify the anchor point close to where the discovery was by remembering the id of the anchor point, and switch to it from the dropdown menu to recover its position. These are rather quick and hacky solutions. A more sophisticated solution would be to save the location somehow and allow naming and note-taking. The figures below show the word embedding display with id (left) and the dropdown navigation menu (right).



The next significant feature Pranov proposed was to visualize the part-of-speech tags. Since the word embedding already has its part-of-speech in the string value, it was simply mapping all the part-of-speech to different colors, and adding a button to switch between different coloring modes. The figure below shows different parts of the same dataset illuminated as filtered by part-of-speech tags.



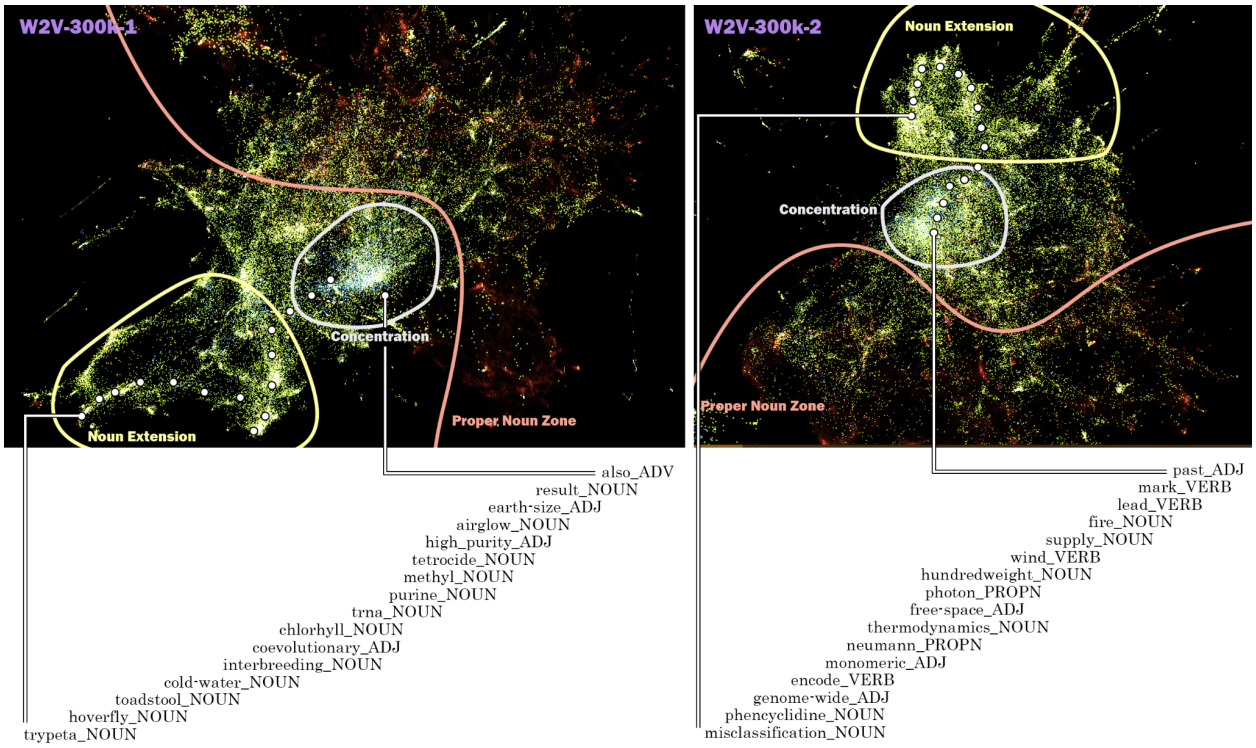
Adam Smith: “I wish I could know that the structure is somewhat stable.”
Fast forward to late summer of 2021, I started taking my interest in humanities seriously and decided to pursue it in the upcoming academic year. But I did not want all the work going into this project gone to waste. So I tried to write this work into my master thesis. Adam Smith, being familiar with embeddings and data visualization, was a natural pick for my second reader.

One of the biggest concerns raised by Adam was regarding the stability of the dataset itself. The original word embedding was provided in 300 dimensions. To get a scatter plot representation in 3D dimension, a dimensionality reduction algorithm called UMAP was used. The basic idea of

the algorithm is trying to preserve local distance relations between points by putting them closer together in the lower dimensional space. This method usually sacrifices global relationships in order to preserve close neighbor relations. The concern regarding the stability of the dataset is that the operation of UMAP is stochastic — the output result is different each time. This is a major threat of validity to any claim from our MCPM probe results in the lower dimension. We need to identify some invariance across different outputs to show that some of our discoveries map onto the original dataset as well.

The work followed was not major features added to the visualization tool, but rather using the tool extensively to identify structures across multiple runs. I ran the UMAP reduction on the same dataset under the same parameter and added an additional dropdown menu so the tool can switch between the two different datasets. The result of comparing the two datasets became a big subsection in my master thesis. This is framed as not only finding the consistent patterns in the reduced dataset, but also demonstrating the usefulness of Polyglot.

The below figure shows one of our discoveries in two different runs (W2V-300k-1 and W2V-300k-2). The global structure of UMAP-reduced dataset can be roughly divided into three different sections: Concentration, Noun Extension and Proper Noun Zone. The Concentration is a high density of commonly-used words. It consists of words with a variety of part-of-speech categories like noun, verb, adjective and adverb. The Proper Noun Zone consists of a wide spread of proper nouns (colored orange but dimmed in the figure), from location names, to scientist names and comic book villains. The Noun Extension consists of a large area of nouns (with some adjectives and adverbs) separate from the Concentration. One of the interesting features is that the words in the Noun Extension zone are much more scientific. The visualization marks a path of samples, going from the Concentration to the Noun Extension. As one can see, the word transitions from commonly used words to specialized words. Identifying structural invariance strengthens the observations made by Slime Mold in the low-dimensional word embeddings, which I will briefly talk about in the **Conclusion in Master Thesis**. The word embeddings in the reduced dimension have a degree of consistency that I argue is present in the original dimension, rather than purely artifacts created by dimensionality reduction.



Conclusion in Master Thesis

In the **Word Embedding** section, I point out that information retrieval is not salient in lower-dimensional representations because of loss of information during dimensional reduction. In my master thesis, I use cosine distance in the original dimension as the benchmark, and compare the structural identification result by MCPM to the result by euclidean distance. I find that the structure identified by MCPM agrees with the benchmark Cosine result more than Euclidean. This shows that UMAP utilizes both euclidean distance as well as connectivity between data to conduct dimensionality reduction, both of which MCPM is sensitive to.

Besides enabling a way to visualize structure more similar to the original representation, the visualization tool also helps identify invariance, thus strengthening the claims in the previous paragraph, because it shows that there is consistent structure in the low-dimensional data that is preserved through the dimensionality reduction process.

Bibliography

[1] Jeff Jones. Characteristics of pattern formation and evolution in approximations of physarum transport networks. *Artificial life*, 16(2):127-153, 2010.

[2] Joseph N Burchett, Oskar Elek, Nicolas Tejos, J Xavier Prochaska, Todd M Tripp,

Rongmon Bordoloi, and Angus G Forbes. Revealing the dark threads of the cosmic web. *The Astrophysical Journal Letters*, 891(2):L35, 2020.

[3] Sunil Simha, Joseph N Burchett, J Xavier Prochaska, Jay S Chittidi, Oskar Elek, Nicolas Tejos, Regina Jorgenson, Keith W Bannister, Shivani Bhandari, Cherie K Day, et al. Disentangling the cosmic web towards FRB 190608. *arXiv preprint arXiv:2005.13157*, 2020.

[4] Cedric Villani. Optimal Transport: *Old and new*. Springer, 2009.

[5] Gabriel Peyre, Marco Cuturi, et al. Computational optimal transport: With applications to data science. *Foundations and Trends® in Machine Learning*, 11(5-6):355-607, 2019.

[6] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In Advances in Neural Information Processing Systems, pages 3111–3119, 2013.

[7] Andy Coenen, Emily Reif, Ann Yuan, Been Kim, Adam Pearce, Fernanda Vi´egas, and Martin Wattenberg. Visualizing and measuring the geometry of BERT. arXiv preprint arXiv:1906.02715, 2019.

[8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805, 2018.

[9] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. OpenAI Blog, 1(8):9, 2019.

[10] Anna Rogers, Olga Kovaleva, and Anna Rumshisky. A primer in bertology: What we know about how BERT works. arXiv preprint arXiv:2002.12327, 2020.

[11] Gregor Wiedemann, Steffen Remus, Avi Chawla, and Chris Biemann. Does BERT make any sense? Interpretable word sense disambiguation with contextualized embeddings. arXiv preprint arXiv:1909.10430, 2019.

[12] Yongjie Lin, Yi Chern Tan, and Robert Frank. Open sesame: Getting inside bert's linguistic knowledge. arXiv preprint arXiv:1906.01698, 2019.

[13] John Hewitt and Christopher D Manning. A structural probe for finding syntax in word representations. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pages 4129–4138, 2019.

[14] Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction, 2018.

[15] Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction, 2018.