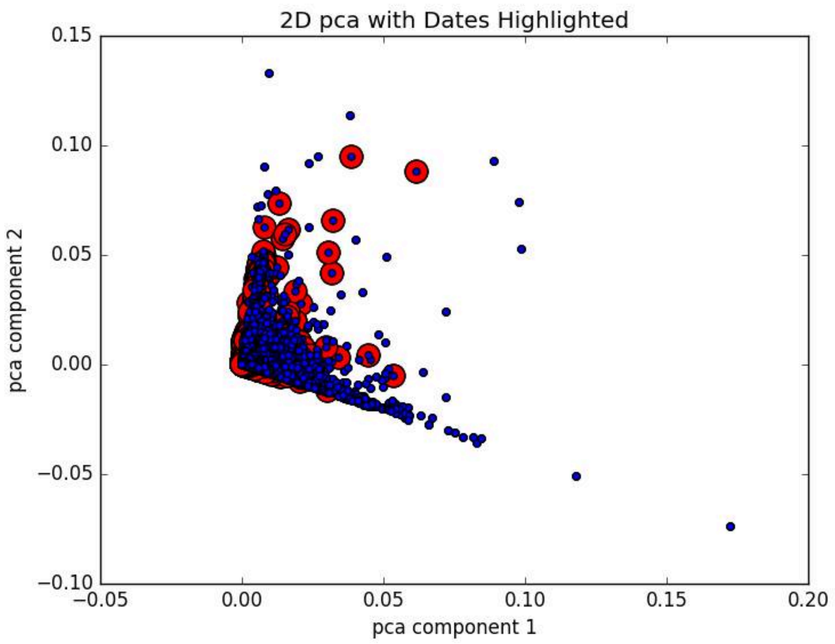
Building an AI to Play the Wiki Game

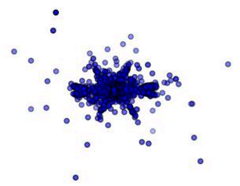
Zack Swafford and Alex Barron

We set out to build an AI that would be capable of playing the Wikipedia Game to the level of a human player. This game involves traversal of the graph formed by websites at wikipedia.org from a random start node to a specific predetermined goal.

Our strategy included

- Indexing simple.wikipedia.org
- Building a state-based search model for the graph
- Applying various simple search algorithms
- Building a feature extractor using NLP to tokenize each page
- Modeling the pages as sparse feature vectors
- Running various ML methods to learn to predict the link-distance between two states
- Applying the best predictor as a (non-consistent) heuristic for A*
- Running K-Means to cluster the states to improve feature vectors
- Applying PCA to visualize the state-space and search results



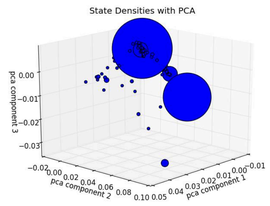


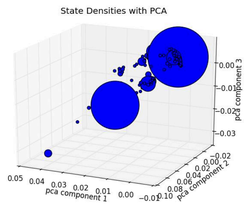
Comparison of ML Algorithms

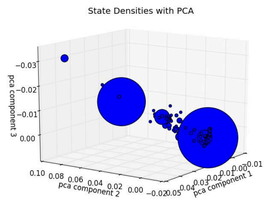
Model	Accuracy	Distance	Wrong 'Unreachable'
One vs. Rest Logistic Regression	55%	.54	3.4%
Multinomial Logistic Regression	59%	.50	1.2%
SGD with Hinge Loss	21%	1.2	0%
SGD with Perceptron Loss	5.8%	1.8	0%
SVM with Hinge Squared Loss	44%	.64	2.4%

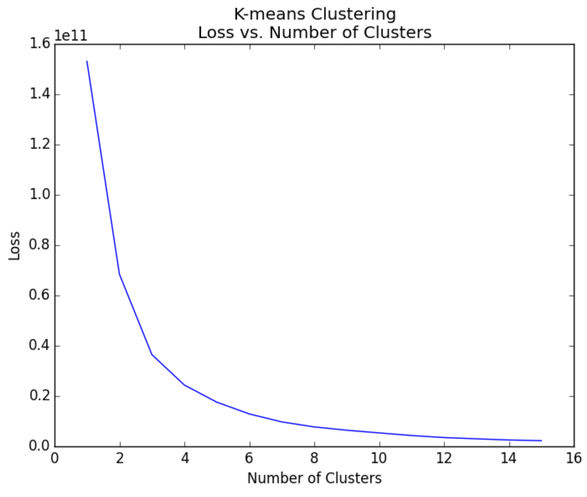
PCA yielded several valuable insights about the data.

- Certain pages were disproportionately more useful in achieving a specific goal article and often one specific final path of two or three articles would be used in the majority of shortest path solutions
- Date articles (e.g. "2007" OR "12 January"), which intuitively connect sporadic ideas, proved to be a very common route to the goal for the minimum path (as represented by the large blue circles in the plots below)
- They also seemed to be outliers in the feature space, so we scatter plotted the data in 2D, highlighting the dates in red. This confirmed that a large portion of the outliers were date articles, suggesting that date articles have a distinct structure which often allows for more minimal paths to the goal state
- On investigation, the remainder of the outliers in the 2D plot turned out to be almost entirely list articles and oddly formatted administrative Wikipedia articles









K-Means was useful insofar as it

- Demonstrated that the data could be clustered under our feature extractor
- Showed that the optimal number of clusters was K=4
- Yielded new features which could be added to the feature vector to improve the accuracy of state classification

Our attempts to examine the content of the clusters to discern their different natures were fruitless, which bespeaks either a deeper significance than we could glean with text analysis or random data.

Comparison of Search Algorithms

Search Type	Path Length	States Explored	Time
Human	7.8	7.8	190 sec
DFS	4.7	210,000	5.2 sec
BFS	4.7	210,000	4.6 sec
UCS	4.7	19,000	.45 sec
A*	5.2	510	12 sec

Although there were many facets to this project, not all were equally illuminating. Acquiring the data and running NLP software on it to featurize it, once the most difficult part of this type of project, was exceedingly simple. Even running a variety of ML algorithms on the data once it was modeled was not hard; this field is well explored. More interesting was modeling the search problem and using the learned predictor as a heuristic for A*. This felt like a tangible application which actually moved toward our stated goal of building a human-level AI for the game. The other unsupervised learning was also very illuminating. Both PCA and K-Means, but particularly PCA, offered valuable insight into the data and problem, and helped us to both visualize and understand the model. With the help of these methods, we accomplished our goal of bringing the AI to close to human-level state exploration (at the cost of the guaranteed minimal path).