1 Background

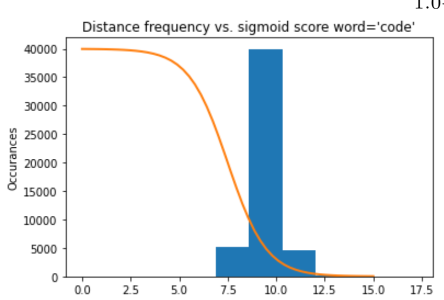
For this project, I decided to approach the 'guessing' segment of the game Codenames. In the setup phase of Codenames, players split into two teams and randomly select 25 words to be included in the current grid. Each team will have a set of these words whose 'selection' means a victory. Once the words have been setup, the players split into two role. One role is the "Codemasters." There is exactly one Codemaster on each team, and they are responsible for giving their teammates hints (a single word and a number of targets) which will lead their teammates to 'selecting' the desired cards. The other role is the "agents." This group is responsible for deciphering the hidden meanings behind the Codemasters' hints, then selecting the correct cards.

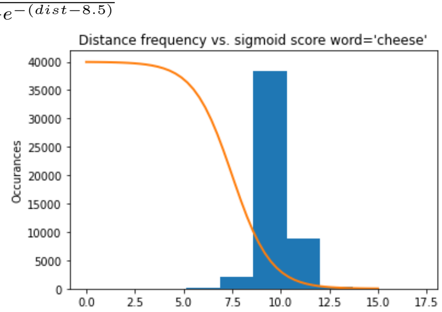
2 GloVe

One talent of the human mind is its ability to quickly find connections between words. Given the strings "cheese" and "lettuce", you are quickly able to notice that they are both food products. Getting a computer to recognize these connections requires a bit of work, but can be done. One way this task can be accomplished is through the use of Global Vectors for Word Representation, or "GloVe vectors". These vectors represent words in a 300 dimensional space, then try to place similar words close to each other, and dissimilar words far from each other. Using these vectors, we can have the computer valuate the similarity of words. While the GloVe distance is not a perfect representation of similarity, it does help us to identify good candidate words.

One limitation of GloVe vectors is that they only provide a raw 300 dimensional distance, and not a clear percentage match. To correct for this, I had to find a function which would rank similar words very highly (close to 1), but rank dissimilar words lowly (far from 1). From my research (shown below) I found that distances above 7.5 meant there was limited similarity, but distances below 7.5 meant there was some real similarity between the words. Below we can see graphs showing frequency of the different distances, and the corresponding confidence percentage.

$$1.0 - \frac{1.0}{1.0+e^{-(dist-8.5)}}$$





From these graphs, we can see the the majority of word comparisons are given very low scores, and that only the very closely related are considered close under this metric. For example, two unrelated words, "Microphone" and "Tree" are given a similarity score of 27%, but two similar words "hand" and "glove" are given a similarity score of 80%. When trying to determine which guess is best, we can rank our options by semantic similarity, then print out the top N, where N is the number of guesses to make.

3 Wikipedia

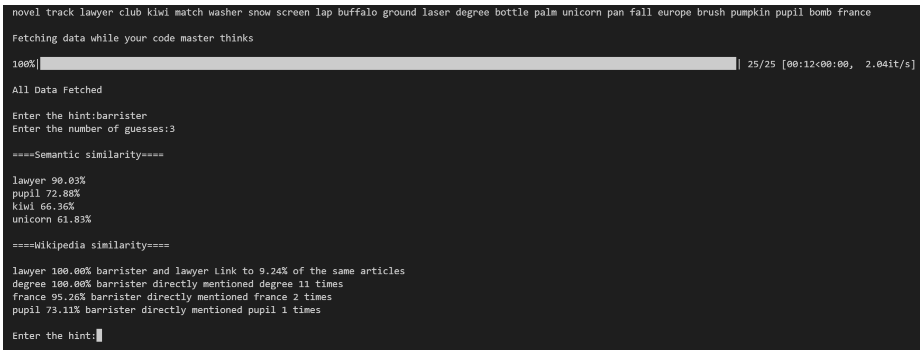
If GloVe distance was a perfect metric, we would have no need for additional computation, however I have not found that to be the case. Some word pairs which we would expect to score very high end up scoring poorly. For example, the pair 'ncis' and 'crime' get a score of only 44%. To remedy Glove's ignorance of pop culture, we can scrape through wikipedia as an additional source of word similarity. The algorithm I derived works something like this. Firstly, we run a word search on Wikipedia and fetch the top K pages matching each word. For my experiments, I used a K of 2 to reduce web requests, but this could be cranked up to ensure no references are missed. Once we have grabbed the top K pages, we look through the words on the fetched pages to see if the pages from one word directly mention the other word. If there is a direct mention, we give the pair a very high score (2 * number of direct mentions). This computation covers the case where two pages are very similar and directly talk about each other.

If there are no direct mentions, we look through the mentioned words of each page and try to find synonyms of the target. To find synonyms, we run each pair through the GloVe matrix to find distance, then through a sigmoid (similar to the previous one) to scale the score from 0-1. To better understand this, consider the pair 'bike' and 'wheel'. Maybe the pages for bike don't directly mention wheel, but they do mention tire. Using this round of checking, we can identify 'tire' and 'wheel' as a close match and assign a moderately high score to this pair. This also prevents scenario where the plural of a word is used (i.e. hand vs hands) which causes direct matching to fail.

Once we have thoroughly combed through the text of the pages, there is still one additional distance metric to consider. Wikipedia can be thought of as a graph, and we can use the links from pages to determine if two words are similar. Image the pair "Chihuahua" and "Poodle". This pair may not mention each other, but they are both dogs and link to a lot of the same dog related pages. We can capture this relation by calculating the percentage overlap between the links of two pages. In cases where they link to 5% or more of the same pages, I award points based on the number of shared links. The final score of each word is ultimately ran through another sigmoid to normalize and provide a confidence percentage that is consistent between plays.

4 In Practice

Initially, I tested my tool by randomly selecting 25 words, then finding a hint and seeing if the computer could reverse my hint well. After testing extensively, I was convinced that my tool was reasonably effective. Below is a screenshot showing the program in action. The words available are visible at the top (novel, track, lawyer) and the hint is visible just below the white bar (barrister). As you can see, it generates two reports, a semantic similarity using GloVe vectors, and a Wikipedia similarity using Wikipedia and GloVe.



With initial testing complete, I got a few friends together to play a real game. I played according to the 'A.I.'s suggestions, and it works surprisingly well. I ended up winning with a 2 card lead. During the game I realized that this was something legitimately cool that other people might want to use. To allow global access, I created a Discord bot which can be accessed by anybody and used to make great guesses. Below is a screenshot of me using my bot in a public discord server. This bot can be added to any discord server using This Link (continued hosting of this project is not guaranteed)