Connections Solver: A Semantic Word Categorization Model

Arunima Gupta: gupta.aru@northeastern.edu
Yulan Wang: wang.yula@northeastern.edu
Yuliya Buturlia: buturlia.y@northeastern.edu
Yash Jayaprakash: j.y@northeastern.edu

Abstract

The popular New York Times game Connections tests a player's ability to group words into categories on the basis of semantic similarity between words. Players are presented with a grid of sixteen words with the objective of assembling them into four overarching categories with four words each. Using a Hidden Markov Model, the Viterbi Algorithm, and word-vector comparisons, we attempt to construct a model that can assign each word of a given grid to a category, utilizing semantic similarity between words and categories. We coin this as a "twist" on the New York Times Connections game.

1 Introduction

Second only to Wordle, Connections is one of New York Times' most played online word games. To win, players must categorize each word from a four by four grid containing sixteen words into the correct category, resulting in four distinct groups with four words each, within the overarching limit of four errors. A typical game of Connections varies in complexity for its groupings, beginning at basic similarities (e.g. suffixes or prefixes) and occasionally ending in non-semantic groupings that rely on a form of abstract deduction to create the category (e.g. the category "One in a Dozen" which consisted of egg, juror, month, and rose).

Previous experiments have been conducted to verify the capability of modern Natural Language Processing systems to solve Connections. In a study presented at the IEEE 2024 Conference on Games, two approaches were explored: Large Language Models (LLMs) from OpenAI, namely GPT-3.5 and GPT-4, and sentence embedding models such as BERT, RoBERTa, MPNet, and MiniLM. Results revealed that GPT-4-TURBO had the high correct matching rate at 29.2%, outperforming GPT-3.5-TURBO with 6.43% success and the best sentence embedding model MPNET with 11.6% success. Notably, correct initial guesses were strongly correlated with higher success rates, while incorrect or nearly correct first guesses often led to failure [Tod24].

For our approach, we decided to focus on semantic similarity in the Connections game. To remain in the scope of the project, we altered the Connections game to focus on matching words to categories, as opposed to words matched to other words based on semantic similarity. We also eliminated error tolerance, relying only on first guesses as the final decision. With this alteration, it became clear that our approach would be most optimal when relying on a Hidden Markov Model [Seg97] that utilized Word2Vec to analyze semantic similarity [Jat19]. The categories are not directly observable to the agent, and the agent needs a way to go from one category to the next, so a Hidden Markov Model has the necessary components to properly represent the game.

2 Methods

For each word given to our model, we begin by applying Word2Vec to obtain the embedding vector. This is then utilized to calculate the cosine similarity between this vector and the vector of each given category. These similarities are preprocessed and stored in an emission matrix to represent the likelihood of each observation being associated with a state(category). For each state, the similarity scores are normalized so that they add up to 1 across all observations for this particular state. We do this by taking the cosine similarity between a word and category and dividing it by the sum of cosine similarities between all other observations and categories. For example if calculating the emission value for $P(\text{"}Strawberry\text{"}|\text{"}Fruit\text{"})$, we would take the cosine similarity between strawberry and fruit and divide it by the cosine similarity between "Basketball" and "Fruit", "Winter" and "Fruit", etc. This ensures that each row of the emission matrix corresponds to a valid probability distribution; in the emission matrix, (number of states $\times$ number of observations), each entry $E[i, j]$ corresponds to the probability of observation j being associated with state i.

Next, we calculate a transmission matrix to represent the probability of transitioning from one state to another. Unlike the emission matrix, this is a (number of states $\times$ number of states) matrix where each element $T[i, j]$ refers to the probability of transitioning from state i to state j. Initially, we attempted to utilize a 16×16 matrix that mapped probabilities of going from one state (ex. Fruit1) to another state (Season1). This was so that each observation would easily have one end state so we could easily enforce the constraint of four words for every category, internally. However, we arrived to the conclusion that representing the states as Fruit1, Fruit2, Fruit3, Fruit4, etc, limited the capabilities of our model. We would not be able to effectively represent if a state was already filled in our transition matrix, without updating our matrix. We even thought about representing the multiple states our game could take in matrix form. We thought we could take a 1×4 array and use it to represent the number of slots filled for each category. For example if our categories were "fruit", "cuisine", "sport", and "cuisine", and we just classified a "strawberry" as a "fruit" we could represent this with an array such as [1, 0, 0, 0]. If each index of the array can go to a maximum of 4, that would imply 4^4 or 256 game states. Meaning we would have a 256 by 256 transition matrix. This would make things quite complicated and it was not clear how we could easily use this with the our Viterbi algorithm implementation. So for the sake of simplicity, in our final solution, we kept a uniform 4×4 transition matrix and assigned each transition probability to one divided by the number of states to ensure an equal probability of transitioning between states. With four categories, the probability is $\frac{1}{4}$ for each element $T[i, j]$.

Having calculated the emission and transition probability matrices, all fundamental components of a Hidden Markov Model (HMM) are now satisfied: the initial distribution $P(X_1)$ represented by the observations array, the transition probabilities $P(X_1)$ represented by the transition matrix, and the emission probabilities $P(O_t|X_t)$

represented by the emission matrix. Notably, HMMs operate under two integral Markov assumptions, the first of which, Temporal Dependency, holds that future states are dependent exclusively on the present state, or the probability distribution of the next state is only dependent on the current state. This assumption is held true, as the probability of matching the next category depends only on the current game state in terms of available words to match and number of categories that can still be filled. The second assumption, Observational Independence, holds that the observation at any given step is independent of all previous and future observations and states, with the exception of the current state. The observations are calculated by calculating the cosine similarity of the Word2Vec vector representations of each word and potential category, so they are not dependent on any particular observation or state in neither the future or the past.

To find the appropriate state (category) for each observation (word), we apply a modified Viterbi Algorithm, extracting the most likely hidden-state sequence. Following the trellis equation for the Viterbi algorithm,

$$\delta_t(S_i) = \max_{j=1}^{N} \left[\delta_{t-1}(S_j) P(S_i|S_j) P(o_t|S_i) \right] \tag{1}$$

with base case

$$\delta_1(S_i) = \pi(S_i) P(o_1|S_i) \tag{2}$$

and final probability

$$P(q) = \max_{i=1}^{N} \delta_T(S_i) \tag{3}$$

where q is the most likely hidden state sequence after T time-steps [Ven24].

Following these formulas as a baseline, we formulated the algorithm to take the maximum of all probabilities at each iteration to find the most probable sequence of states. Additionally, backtracking guarantees that the best global solution, beyond simply local, is found. Our final categorization strategy is summarized in Algorithm 1.

Algorithm 1 Viterbi Algorithm with Constraints

Input: Observations, States, Emission Matrix, Transition Matrix

Output: Best State Sequence

Initialization:

Create DP table `dp` of size `(n_states` × `n_observations)`

Create Backpointer table `backpointer` of the same size

foreach *state s in States* **do**

  `dp[s][0]` ← $\frac{1}{n\_states}$ · `Emission_Matrix[s][0]`

  `backpointer[s][0]` ← −1

Recursion:

foreach *time step t from 1 to* `n_observations - 1` **do**

  foreach *state s in States* **do**

    `probabilities` ← []

    foreach *prev_state ps in States* **do**

      `probabilities.append(dp[ps][t-1]`

      × `Transition_Matrix[ps][s]`

      × `Emission_Matrix[s][t])`

    `dp[s][t]` ← max(`probabilities`)

    `backpointer[s][t]` ← arg max(`probabilities`)

Backtracking:

`best_state_sequence` ← []

`best_state` ← arg max(`dp[:, -1]`)

`best_state_sequence.append(best_state)`

foreach *time step t in reverse order from* `n_observations` *- 1 to 0* **do**

  `best_state` ← `backpointer[best_state][t]`

  `best_state_sequence.append(best_state)`

Return *reversed(best_state_sequence)*

3 Data and Experiments

3.1 Data Source

Our primary source of data is contained in "word2vec-google-news-300". This particular model contains pre-trained vectors trained on the Google News dataset of about one hundred billion words. The model has 300-dimensional vectors for three million words, as well as phrases. For each necessary word and category, we extracted the associated vector from this model.

3.2 Experiments

3.2.1 Semantic Similarity

To test the base capabilities of the model, sorting common words by semantic similarity to the category options, we used seven potential categories, fruits, clothing, colors, sports, flowers, shapes, and cuisines, each with twenty associated words. For example, fruits were represented as such:

fruits = ["blueberry", "strawberry", "kiwi", "apple", "banana", "orange", "mango", "grape", "pineapple", "watermelon", "papaya", "cherry", "peach", "pear", "plum", "guava", "lychee", "fig", "apricot", "dragonfruit"]

For each of 10,000 trials, four categories of the provided seven were randomly selected. For each chosen category, four of the twenty available words were selected randomly to create the required sixteen word Connections-like grid. For each, it was ensured that each category and word were unique, modeling after the Connections game. After calculating the appropriate emission and transition matrices, our algorithm was then run on each generated grouping of observations, states, emission matrix, and transition matrix. The accuracy of the model was calculated with the following equation on each iteration, obtaining the number of correctly categorized words by checking if the given state for each word in the best path given by the model truly contained this word in its previously defined array.

$$Accuracy = \frac{\text{correctly categorized words}}{16} \times 100 \qquad (4)$$

3.2.2 Homonyms

Having confirmed a degree of accuracy for simple word to category semantic analysis for the model, we shifted focus to more ambiguous and complicated categories and word options. One such test included the following two categories and potential corresponding sixteen words with the purpose of testing the model's accuracy with homonyms.

tree = ["palm", "limb", "trunk", "crown", "heart", "veins", "skin", "roots", "branch", "sap", "knot", "leaf", "ring", "bark", "shoot", "stem", "canopy", "twig", "seed", "flower"]
human = ["head", "arm", "leg", "torso", "hand", "foot", "eye", "ear", "nose", "mouth", "heart", "lungs", "skin", "brain", "stomach", "bones", "fingers", "toes", "hair", "teeth"]

Each word within the human category can have an interpretation that places it in the tree category. These examples are modeled in Table 1 below.

To test the model's capability to correctly categorize with metaphors and homonyms, we ran three trials. The first used the tree category in each of the 10,000 trials along with 3 other random categories, excluding human. The second trial used the human category in each of the 10,000 trials with three other random categories as well, excluding tree. Finally, we ran the third trial with both the tree and human categories with two additional random categories.

Homonyms Between Body and Tree		
Word	Human Interpretation	Tree Interpretation
Palm	inner part of the hand	tropical tree
Limb	arm or a leg	large branch
Trunk	torso	main woody stem of a tree
Crown	top part of the head	upper part of a tree
Heart	organ that pumps blood	central part of a trunk
Veins	blood vessels	vascular structures in leaves
Skin	outer layer of the body.	bark
Roots	metaphorically refers to ancestry	underground part of the tree
Branch	metaphorically, nervous system or family tree	limb growing from the trunk
Sap	metaphorically, energy or vitality	fluid in the vascular system

Table 1: Homonyms Between Body and Tree

3.2.3 Connotations and Emotions

Finally, we tested the model's ability to differentiate words with various connotations from both each other and from other simple words. To do so, five additional categories were created: positive, negative, weather, virtues, and flaws. Additionally, color was also considered a category with connotation due to its common use in figurative language. The positive category contained twenty words with positive emotions, such as "peace", "gratitude", "comfort", and "delight", while the negative contained the opposite with words such as "envy", "anxiety", "frustration", and "dread". Weather contained common weather phenomena, such as "storm", "hail", "hurricane", and "sunshine". Virtues and flaws were identically formatted with words akin to "wisdom", "honesty", "integrity", and "loyalty", and "dishonesty", "laziness", "impatience", and "cruelty", respectively.

Using these new connotation-imbued words and categories, we ran five trials. One trial with 10,000 iterations was a control with four randomly chosen simple categories as used previously and no words from the newly defined connotation list. The following progressively reduced the number of simple categories randomly chosen while increasing the number of randomly chosen categories with clear connotations, while holding all other parts of the trials constant. Thus, we ran a trial with 3 simple and 1 with connotation, 2 simple and 2 with connotation, 1 simple and 3 with connotation, and 0 simple and 4 with connotation, and we analyzed the results.

4 Results

For the simple collection of seven categories with four randomized categories and sixteen randomized words over 10,000 trials, the model matched the word to the correct category 91.439% of the time, as visualized by the following Figure 1 depicting the percentage of correctly classified words for each randomized trial.

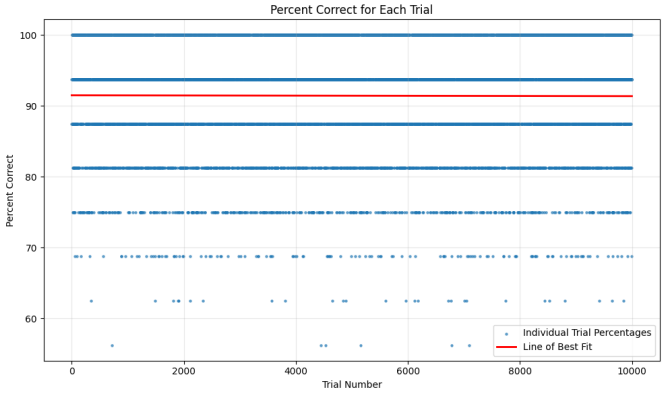

Figure 1: Percentage of correctly categorized words over 10,000 trials.

We then explored the trials in homonyms as previously outlined, starting with 10,000 trials for three random categories, with the exception of the human category, and the tree category. This resulted in an accuracy of 84.317% as visualized in Figure 2.

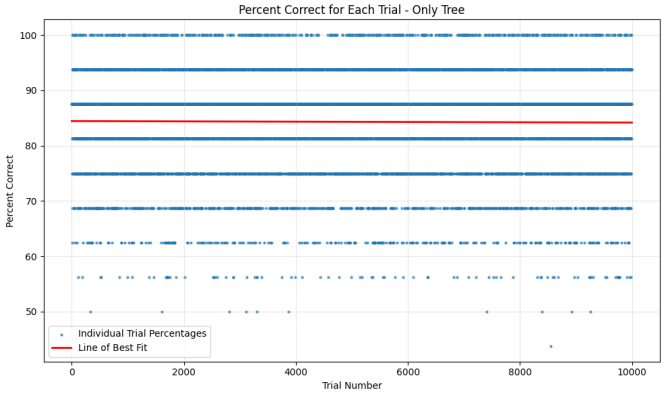

Figure 2: Percentage of correctly categorized words over 10,000 trials, with tree and three randomly selected categories.

This was continued with isolating the human category with 10,000 trials for three random categories, with the exception of the tree category, and the human category, resulting in an accuracy of 88.339% as visualized in Figure 3 below.

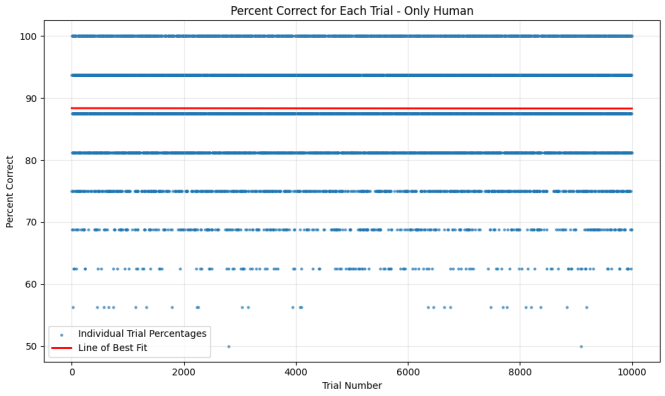

Figure 3: Percentage of correctly categorized words over 10,000 trials, with human and three randomly selected categories.

Finally, we combined the aforementioned categories to explore the model's capability of differentiating between the tree and human words, which resulted in an overall accuracy of 79.534%, as visualized below in Figure 4. This shows a decrease in the model's ability to differentiate between categories, and thus unveils a weakness when it comes to homonyms and more complex, ambiguous words. At nearly 80%, this categorization is still fairly successful, and thus does not significantly retract from the efficacy of the model.

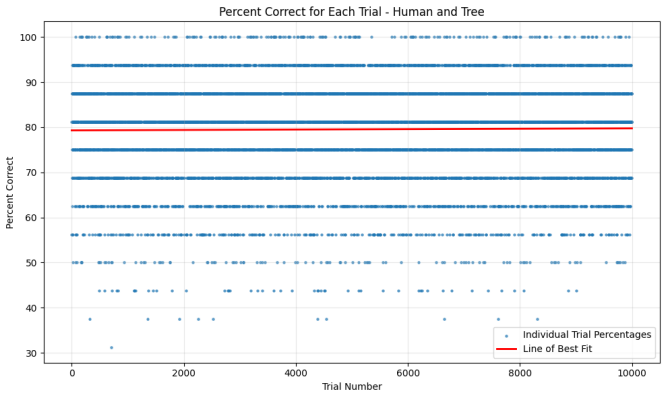

Figure 4: Percentage of correctly categorized words over 10,000 trials, with human, tree, and two randomly selected categories.

Finally, we explored the model's capability to interpret connotation, using it to classify words. Using the previously developed experiments with 10,000 trials for each variation, we obtained the following results in Table 2.

Percent Correct Dependent on Connotation Categories		
Simple Categories	Connotation Categories	Percent Correctly Classified Words
4	0	91.47625
3	1	91.806875
2	2	89.643125
1	3	86.1175
0	4	81.37125

Table 2: Percent of Words Classified Correctly for Each Variation of Number of Connotation Categories

To show the percentage of correctly classified words for each category for each individual trial, the graphs below, Figures 5, 6, 7, 8, 9, visualize these points for each variation in trials. We can observe that the percentage of correctly classified words decreases with the addition of categories with notable connotational similarities, with the exception of the trials with three simple categories and one connotation category. This trial had greater accuracy than four simple by a very minimal amount, which could be attributed to the scope of our categories and variety in word options for each trial.

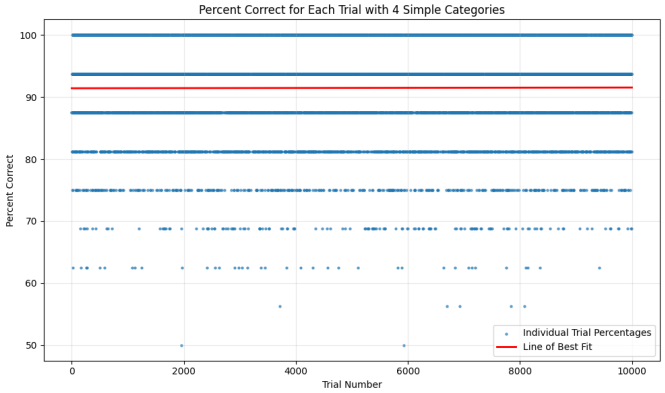

Figure 5: Percentage of correctly categorized words over 10,000 trials with only simple categories for control.

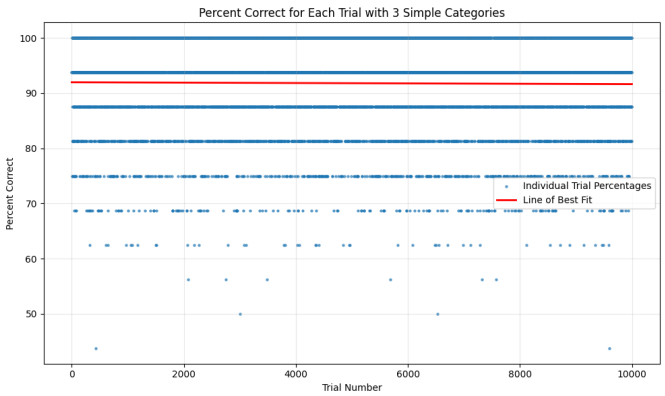

Figure 6: Percentage of correctly categorized words over 10,000 trials with 3 simple categories and 1 with strong connotation.

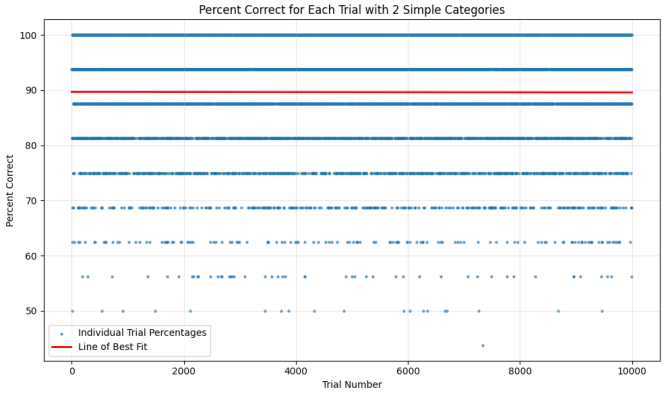

Figure 7: Percentage of correctly categorized words over 10,000 trials with 2 simple categories and 2 with strong connotations.

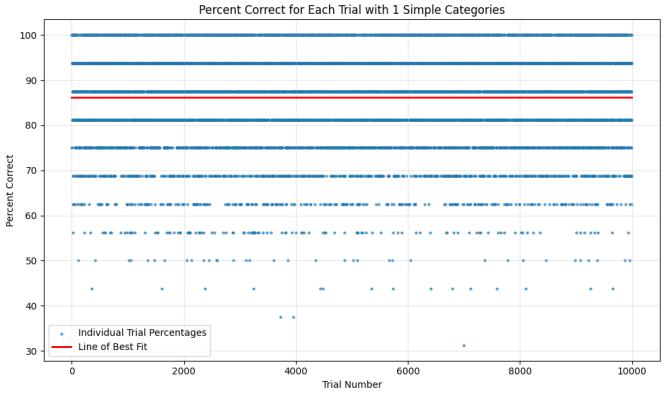

Figure 8: Percentage of correctly categorized words over 10,000 trials with 1 simple categories and 3 with strong connotations.

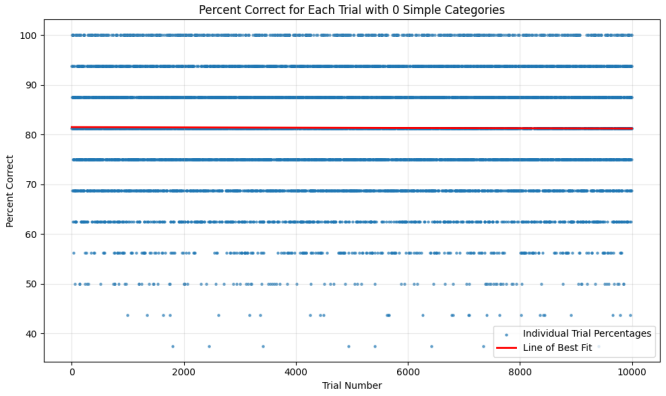

Figure 9: Percentage of correctly categorized words over 10,000 trials with no simple categories and 4 with strong connotations.

5 Conclusion

We find that our model demonstrates a promising approach to solving a modified Connections game by leveraging semantic similarity through Word2Vec embeddings and a Hidden Markov Model with a modified Viterbi Algorithm. The model has a notably high accuracy in classifying words to categories based on semantic similarity at around 91% of tested words categorized correctly and a fairly high accuracy with more complex word groupings with around 80% of words categorized correctly. We assert that Word2Vec embeddings provide reliable calculations for emission probabilities for word-category classification. Additionally, a Hidden Markov Model coupled with the Viterbi Algorithm is highly effective for sequential categorization, despite scenarios when the given observations involve ambiguity or overlapping meanings, as shown with homonyms. While the model performs well on homonyms as a whole, accuracy declines when numerous categories contain semantically or morphologically overlapping terms, suggesting areas for improvement. Additionally, analysis on semantic similarity on the basis of connotation demonstrates a satisfactory level of preliminary success. Future work could include expanding to support non-semantic categories that rely on abstract or lateral reasoning, making the grid of words more similar to possible Connections boards, as well as runtime and accuracy improvements.

6 Link to GitHub Repo

`https://github.com/arunimag23/aiproject2024`

7 Team Contributions Statement

Arunima: emission and transition matrix logic, reworking word2vec, read me, presentation slides (Game Set Up and HMM), methods section contributions for report, project management.
Yulan: Verterbi algorithm, game GUI set up, presentation slides (Viterbi algorithm and To Dos), final model fixes.
Yash: Viterbi Algorithm, transition/emissions matrices, refining GUI, incorporating the final model into the GUI, demo video.
Yuliya: testing/trial and graph creation, refining algorithm for ambiguous categories, final report.

References

[Tod24] G. Todd, T. Merino, S. Earle, and J. Togelius. Missed connections: Lateral thinking puzzles for large language models. *arXiv:2404.11730*, 2024.

[Jat19] D. Jatnika, M. Bijaksana, and A. Ardiyanti. Word2Vec model analysis for semantic similarities in English words. *Procedia Computer Science*, vol. 157, pp. 160–167, 2019. doi:10.1016/j.procs.2019.08.153.

[Seg97] F. Segond, A. Schiller, G. Grefenstette, and J.-P. Chanod. An experiment in semantic tagging using hidden Markov model tagging. *Proceedings of Automatic Information Extraction and Building of Lexical Semantic Resources for NLP Applications*, 1997.

[Ven24] R. Venkat. Hidden Markov Models. *AI Resources*, 2024. Available at: `https://rajagopalvenkat.com/teaching/resources/AI/ch6.html#hmm`.