Creating an Intelligent Word Association Agent for the Game Apples-to-Apples™

Jasvinder Dhaliwal, John Divinagracia, Ayden Hogeveen

CMPT 355: Introduction to Artificial Intelligence / AS01

March 31, 2024

Course Instructor: Dr. Calin Anton

Context of the Paper

1. We started working on this paper on March 19, 2024
2. We sought support from MacEwan University Library: N/A
3. We sought support from the Writing Centre: N/A
4. We provided or received peer editing or peer support: N/A

Abstract

This paper presents our approach to creating an intelligent word association agent capable of playing the card game *Apples-to-Apples™*. We were inspired by the 2025 EAAI Mentored Undergraduate Research Challenge set by the Journal of Engineering Applications of Artificial Intelligence. Numerous researchers have delved into related problems considering the relationship between artificial intelligence and natural language. These have made use of the power of artificial intelligence in everyday applications such as predictive text, search engines, vocal assistants, etc. We aimed to create a trained model to pick the best red card (noun) and green card (adjective).

Table of Contents

1. Introduction

The challenge proposes that students participate in an artificial intelligence research project in order to create an intelligent agent capable of playing the game *Apples-to-Apples™* [1]. The game involves a player viewing a prompt card and determining the most relevant card in their hand to play, playing the card, and a judge player determining the most relevant card from each player's played card [2]. For this game, an intelligent agent must be able to maximize the performance measure of relevance to the prompt card.

Therefore, our research problem becomes (1) how can the agent define the relevance of a card and (2) how can we determine the defined criteria in choosing a card for the agent to play. In the paper, we detail the methods that we used to define "relevance," the observations we made that led us to change and improve those methods, and the conclusions and future directions that can be taken from this research project.

The increased visibility and priority of natural language processing and large language models in computer science and popular culture make projects like this one an important way for students to familiarize themselves with some of the technologies, methodologies, and practices of machine learning and neural network development.

2. Methods

The methods used to determine the most relevant card to the prompt started very simple and became more complex as we dug deeper into the world of Python natural language processing practices and libraries. Our research process led us to use Python for this project in order to use the ample natural language processing libraries.

2.1 Random Strategy

The first and simplest method used to create an agent that took an inputted green card and returned a red card from their hand was a simple random agent.

Figure 2.1.1

```
def play_random_card(self):
    """
    Plays a card from the agent's hand randomly
    :return: card datatype
    """
    return self.hand.pop(random.randint(0, len(self.hand) - 1))
```

2.2 Alliteration Strategy

The next agent strategy was to choose a card that started with the same letter as the prompt card, if possible, otherwise playing the first card in hand.

Figure 2.2.1

```
def play_alit_card(self):
    """
    Plays a card from the agent's had based on the first letter of the green
    card, otherwise plays the first card
    :return: card datatype
    """
    first_letter = self.green_card[0]
    for i in range(len(self.hand)):
        if (self.hand[i][0] == first_letter):
            return self.hand.pop(i)

    return self.hand.pop(0)
```

2.3 Part of Speech Association Strategy

This agent strategy used the nltk library in Python to identify the part of speech of the cards in hand and the prompt card and playing the card that matched, if possible, otherwise playing the first card in hand.

Figure 2.3.1

```
def play_card_pos(self):
    """
    Plays a card from the agent's hand based on similar parts of speech
    :return: card datatype
    """
    tags = self.get_pos_tags()

    if (self.green_card is not None):
        target = nltk.pos_tag([self.green_card])

        for tag in tags:
            if (target == tag[0][1]):
                return self.hand.pop(tags.index(tag))

    return self.hand.pop(0)
```

2.4 Swap Distance Strategy

The swap distance strategy found the best card by determining the most similar card based on the Levenshtein distance* between the prompt card and the red cards in hand and returning the card closest to the prompt card.

Figure 2.4.1

```
def play_associated_card(self):
    """
    Plays a card from the agent's hand based on the Levenshtein distance
    between the green card and the red card. This distance is calculated from
    the number of single-character edits required to change one word into the other.
    :return: card datatype
    """
    card_scores = []
    for i in range(len(self.hand)):
        card_scores.append(nltk.edit_distance(self.green_card, self.hand[i]))

    return self.hand.pop(card_scores.index(max(card_scores)))
```

*Levenshtein Distance is the minimum number of single-character edits to one word to make it into another. In this case, it is from the prompt card to the cards in the agent's hand. [3]

2.5 Model Training Strategy

We train a model by giving it a training corpus (WikiText [4]) and using a Word2Vec algorithm provided by the 'gensim' library. The training creates a dictionary of words to vectors that we can use to get the cosine similarity between cards. Since cards contain multiple words, calculating the average vector of all existing words within the card allows us to get the vector of the card's overall "meaning."

```
size_vec = 300

def train_model():
    sentences = []

    for sentence in brown.sents():
        sentences.append(simple_preprocess(' '.join(sentence)))

    for sentence in reuters.sents():
        sentences.append(simple_preprocess(' '.join(sentence)))

    add_red_card_examples(sentences)
    add_green_card_examples(sentences)

    model = Word2Vec(vector_size=size_vec, window=5, min_count=5, workers=10)
    model.build_vocab(sentences)
    model.train(sentences, total_examples=model.corpus_count, epochs=30)
    wv = model.wv
```

3. Results

3.1 Example Outputs

To provide an idea of the progress and evaluation of each strategy, we will give several sample outputs from each agent to provide a snapshot of the ability of each agent to find the most relevant card.

Figure 3.1.1: Sample Random Agent Game Output

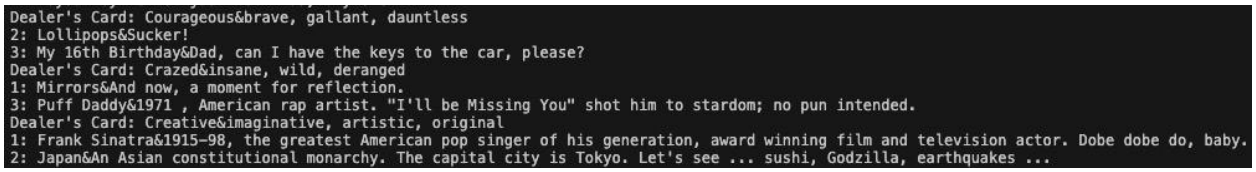


Figure 3.1.2: Sample Alliteration Agent Game Output

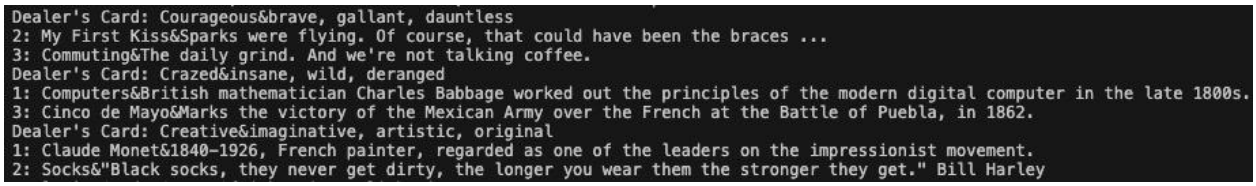


Figure 3.1.3: Sample Part of Speech Agent Game Output

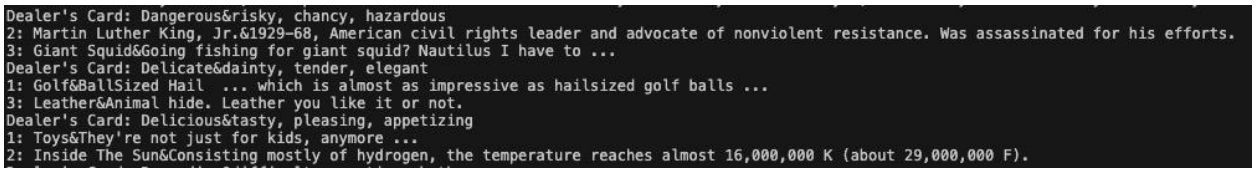


Figure 3.1.4: Sample Levenshtien Distance Agent Game Output

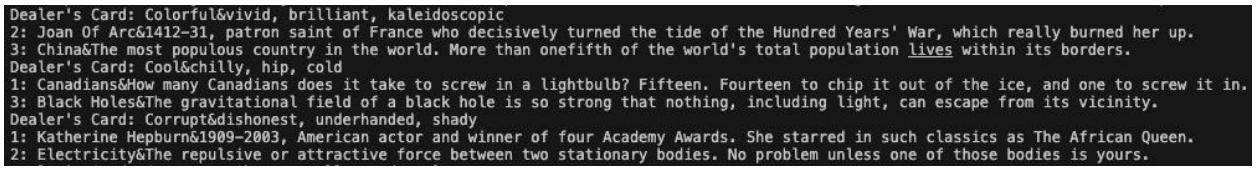
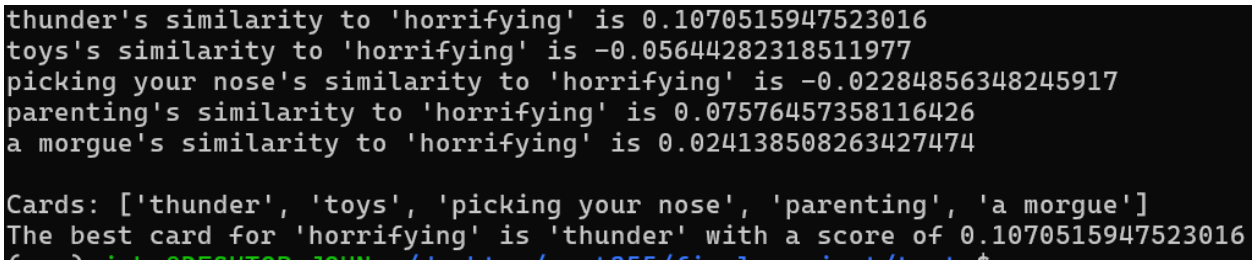


Figure 3.1.5: Sample Model Training Output



4. Simulation

In this next section, we will provide some game results of five trained agents

Interestingly, below, we can see that all trained agents related a person, if available, to the word “Animated.”

```
---------------NEW ROUND------------------
Model player (#1) is playing as a: player

Model player cards in hand:
Card 1: Carl Sagan - 1934-96, American astronomer and pioneer exobiologist. Touched many lives. Billions and billions of
 them.
Card 2: Paris, France - The governmental, cultural and gastronomic capital of France. Or the world. Just ask the French.
Card 3: A Sunrise - "But he who kisses the joy as it flies/ Lives in eternity's sunrise." William Blake
Card 4: Humphrey Bogart - 1899-1957, American film actor, starred in such classics as The Maltese Falcon and Casablanca.
 "Here's looking at you kid."
Card 5: Quentin Terantino - 1963 , Groundbreaking American director and actor. Nice movies, but not nearly enough blood.
Card 6: Going To School - Secondary education? It's elementary.
Card 7: Lucille Ball - 1911-89, American actor and comedian. You love Lucy? I love Lucy.

Green card from judge: Animated - lively, exuberant, spirited

Model player chose the card: Quentin Terantino - 1963 , Groundbreaking American director and actor. Nice movies, but not
 nearly enough blood.

Winner is Player 5!

Other players cards:
Card 1: Quentin Terantino - 1963 , Groundbreaking American director and actor. Nice movies, but not nearly enough blood.
Card 2: Robin Williams - 1952 , versatile American comedian and actor. The furriest man in show business.
Card 3: Oprah Winfrey - 1954 , Television talk show host, actor and allaround American success story.
Card 4: Michelle Pfeiffer - 1957, American actor whose career began with Grease 2. She'd like us to forget that but reme
mber her Oscar nominations.
```

The judge (trained agent) picks “Paying Taxes” is the most similar to “Demanding”

```
---------------NEW ROUND------------------
Model player (#3) is playing as a: player

Model player cards in hand:
Card 1: Berlin - 1945  Adolph Hitler's last stand. Bunker ... I mean hunker down, soldier.
Card 2: Hairballs - Cough it up, kitty ...
Card 3: London - Founded even before the Romans reached England's shores, now the capital of Great Britain.
Card 4: John F. Kennedy - 1917-63, 35th president of the US. "Ask not what your country can do for you; ask what you can
 do for your country."
Card 5: San Francisco - California's City by the Bay. Where many hearts are left.
Card 6: Lightning - Electrical discharge between rain clouds, or between a rain cloud and the earth, or between a rain c
loud and an idiot with a kite.
Card 7: My Future - "The future's so bright, I gotta wear shades ..." Timbuk3

Green card from judge: Demanding - difficult, exacting, bothersome

Model player chose the card: My Future - "The future's so bright, I gotta wear shades ..." Timbuk3

Winner is Player 2!

Other players cards:
Card 1: Picking Your Nose - Noses run in our family.
Card 2: Paying Taxes - Death or Taxes? Do we have to choose right away?
Card 3: My Future - "The future's so bright, I gotta wear shades ..." Timbuk3
Card 4: Silk - Produced as a cocoon covering by the silk worm, and used in fine fabrics and textiles.
```

Word embedding allows for vector addition, as seen in the below example:

Output of the most similar word vector when doing vec(‘woman’) + vec(‘king’) - vec(‘man’)

```
print(wv.most_similar(positive=['woman', 'king'], negative = ['man']))
```

```
[('queen', 0.5682238340377808), ('monarch', 0.5381150245666504), ('empress', 0.5296509
265899658), ('isabella', 0.5059524774551392), ('princess', 0.4858843982219696), ('cons
ort', 0.4753859341144562), ('dowager', 0.47513100504875183), ('ruler', 0.4669325053691
864), ('regent', 0.4657449424266815), ('noblewoman', 0.4557579755783081)]
```

5. Discussion

In this section, we can talk about future work, what we learned about artificial intelligence agents and word association games, natural language processing, and more.

A larger training corpus could improve the performance of our agent, but would require more time and resources for the agent to train on. We could potentially use the 20 GB Wikipedia corps, containing billions of words, or even the OSCAR (Open Super-large Crawled Aggregated coRpus) corpus, which contains over 3TB worth of data.

Our development process went through many stages of testing, checking, doing research and developing the agent at the same time. In future work, research will be of special focus early on. Similarly, the possible avenues of development shall be explored broadly to allow a clear view of which path leads to a comprehensive, complete, and focused implementation.

With the model training strategy, there is motivation to test the statistical significance of each result in a hand– however, there is an obvious roadblock. When using cosine similarity to measure the *strength* of a word association, an external measure is required to conduct statistical analysis (i.e., t-test or p-value). This external measure can either be a human’s judgment or another model’s judgment (i.e., LLM, another Word2Vec model, etc.), which introduces either the bias of a human or the bias of a model. This bias can be understood in the context of the innate subjectivity of Apples to Apples.

6. References

[1] G. Freedman, R. (2025). 2025 EAAI Mentored Undergraduate Research Challenge: Playing Word Association Games. *AI Matters*, *X*(X). https://meskanas.macewan.ca/pluginfile.php/1444969/mod_resource/content/1/MURC_Playing_Word_Association_Games.pdf

[2] Wikipedia contributors. (2024b, February 11). *Apples to apples*. Wikipedia. https://en.wikipedia.org/wiki/Apples_to_Apples

[3] Nam, E. (2021, December 8). Understanding the Levenshtein distance equation for beginners. *Medium*. https://medium.com/@ethannam/understanding-the-levenshtein-distance-equation-for-beginners-c4285a5604f0

[4] WikiText dataset https://huggingface.co/datasets/wikitext

[5] *Gensim: topic modelling for humans*. (n.d.). https://radimrehurek.com/gensim/models/word2vec.html

[6] *Word Embeddings in Python with Spacy and Gensim*. (n.d.). https://www.cambridgespark.com/info/word-embeddings-in-python

[7] Coding Lane. (2022, June 11). *Word Embedding Python | RNN | Detailed explanation* [Video]. YouTube. https://www.youtube.com/watch?v=pFOKxJO4gu8

[8] Coding Lane. (2022b, June 25). *Part 2 | Python | Training Word Embeddings | Word2Vec |* [Video]. YouTube. https://www.youtube.com/watch?v=39w4WSxvRQM

[9] ritvikmath. (2021, June 23). *Word2VEC : Natural Language Processing* [Video]. YouTube. https://www.youtube.com/watch?v=f7o8aDNxf7k

[10] StatQuest with Josh Starmer. (2023, March 13). *Word embedding and Word2Vec, clearly explained!!!* [Video]. YouTube. https://www.youtube.com/watch?v=viZrOnJclY0