CS 4740 Fall 2015 **Due 11:59pm Friday, December 4**
Albert Heidmann **ath55@cornell.edu**

NLP System Design: Cards Against Humanity

Albert (Teddy) Heidmann

1. Problem

The problem I came up with relates to the popular card game *Cards Against Humanity.* The game involves 4^+ players, and each round, the judge draws a fill-in-the-blank 'black card', and each of the remaining players plays a 'white card' from their hand that contains a phrase on it such as **"Overcompensation"**, **"The Underground Railroad"**, **"Teaching a robot to love"**, **"Former President George W. Bush"**, and numerous other mature-content phrases. The judge then selects which of the played white cards they like best, and the person that played that card wins a point. The judge rotates every round, and you keep playing until you get bored, run out of cards in the deck, or (according to the rules) someone gets 6 points.

The NLP task I envision is creating a system that runs each round and performs the following task: given a judge, a black card, and a hand of white cards, the system reads each black and white card pairing, and returns a set of 'features' that that pair of cards falls into. For example, suppose Player A was the judge, and the current black card was **"A romantic, candlelit dinner would be incomplete without ________"**. Player B has a few cards in their hand, and the NLP system would look at each white card in combination with the black card and figure out what the completed phrase should be tagged as. For example, when looking at the black card combined with **"Black people"**, the system should return the feature Racism, because that combination of black and white cards can be seen as racist. If B has **"Lumberjack fantasies"**, the system might return Sex or Absurd, because this combination of cards can be seen in a sexual way, but also in a very strange, unusual way, as lumberjack fantasies are not common (as far as I know).

These features can then be used as part of a machine learning algorithm to figure out each judge's preferences and play the best possible white card, allowing a computer that incorporates this NLP system to win more and more over time as it develops and understanding of what each judge's individual preferences is.

I chose this task because for CS 4701: Practicum in Artificial Intelligence, my partner and I wrote an 'optimum Cards Against Humanity player' which takes a set of manually tagged black and white cards and uses these tags to figure out what a judge's preferences are. My partner had to hand tag all 550 cards in the deck, and having an NLP system would have made the task much easier.

2. General Approach / System Architecture

The NLP system would follow the general architecture described above. The major components would be first getting all the pairs of white cards and black cards, then parsing the text and pulling out the meaning of the phrase, then categorizing each pair into the various feature vectors. Part 1 is very straightforward and just requires putting the white card into the blank in the black card's phrase, then calling a function to return the feature set for that pair of cards. This function is a littler harder to implement, and encompasses parts 2 and 3 mentioned above.

Part 2 would need to use WordNet to find definitions of words and figure out their meaning, and this portion would take the longest amount of time and effort, as Natural Language Understanding is a difficult task. Chapters 9 and 10 of the NLTK book [1] explain the procedures involved in figuring out the meaning of a phrase, and it's definitely a complicated process. This tagged phrase would then be passed into the function for part 3.

Part 3 would use the tags from part 2 and the set of features identified for the cards, then plug these into a clustering algorithm to figure out which categories the phrase is closest to. This allows each phrase to fit under multiple categories, and we return the set of categories that it falls under.

The alternative architecture I considered was what my CS 4701 partner and I ended up implementing, which is tagging the white and black cards at the beginning, then using a machine learning algorithm with these features. This does not take into account the context for a given black card / white card pairing, so the features are pretty much set in stone, and cannot be changed. While this is useful in that there is less computation required (calculate features one time per card for a total of 550 calculations, versus calculate features each round for every card in your hand, i.e. 6 time for every round you play), the lack of context means that it is a lot less effective than the NLP system could be.

For example, given the white card **"Sarah Palin"**, a pre-tagging approach would probably mark it as POLITICS and FEMALE, both of which make complete sense. However, pairing it with the black card **"What does Dick Cheney prefer?"** the predominant feature would most likely be SEX, simply because of the context. With examples like this in mind, I felt the best system would take context into account, and so pre-tagging the cards with features was not the best option.

3. Data and Data Annotation

The corpus for this NLP system is the set of black and white cards. The original set is 550 cards total, with 460 white cards and 90 black cards, but there are a number of expansion packs, holiday sets, and various other promotional games, adding close to 1000 cards more with a similar white : black ratio. For my CS 4701 project, we limited the cards to the original set, which meant that there were fewer possible combinations, and therefore our system could learn what the 'best' cards were much faster and more successfully than with a set of 1500 cards.

This system could also be improved by searching the web for recent articles and political opinions, but this would require a lot more architecture and system overhead, decreasing the overall speed of the system. Also, the system will still work without it, but will function using more on-the-surface features, rather than relating them to current events or history of events. For example, with **"Sarah Palin"**, a search online could provide many useful political opinions, and we would be better able to tag the card based on the context. If Sarah Palin had recently said something regarding religion or violence, then the system could return RELIGION or VIOLENCE, thereby increasing the overall effectiveness of the system, given that the judge is a human and has a similar understanding of current events.

The list of black and white cards is available online for the original deck, and I'm sure that someone somewhere has typed up the list of cards for every expansion pack and other specialized set. The creators might also be interested in the possibility of their work being featured in an NLP system, and due to increased publicity and a possibly better understanding of the game, they might cooperate and share a list of every card they've printed. I already have nicely formatted `CSV` files for the black and white cards of the original deck, so those are very easily imported into a database to be used by the system.

The project won't require any manual annotation of training data, since the task is to extract the information automatically using a semi-dictionary approach. However, it will return a set of features, and these features need to come from somewhere. The features my partner and I used were: SERIOUS, SEX, DISASTER, DISGUST, ABSURD, RELIGION, VIOLENCE, SCIENCE, RACE, POLITICS, HEROISM, CULTURE, SPORTS, FOOD, ARTS, MALE, FEMALE, UNREALISTIC, HEALTH, POVERTY, SELFISH, APPEAL, OTHERS, PERSONALITY, AGE, and ANIMAL. In order for the clustering algorithm to work properly, the system will need to use previous examples of tagged phrases or look up WordNet definitions and synonyms to properly match the phrase to the best feature(s).

4. Methods and System Development

As mentioned above, there are 3 components to this system: parsing and forming the black / white card pairs, pulling out the meaning from the phrases, and tagging these phrases with features. For the first part, it's simply reading in the strings from the database, then iterating through them and calling a function which returns the result of parts 2 and 3 combined. There is no need to implement any additional algorithms for this section.

For part 2, we need to use the Natural Language Understanding methods referred to in chapters 9 and 10 of the NLTK book [1]. There are no algorithms involved other than tagging the words with meanings, which is a very similar task to using the tree system described in class for finding the correct sequence of tags for a sentence.

For part 3, we need to use a clustering algorithm such as K-means clustering in order to match the phrases with the features. For this specific problem, we can take each of the meanings found from part 2 and put them into K-means algorithm in order to find the closet feature(s) for that meaning. We then return this set of features, giving us the solution we want.

5. Implementation

Since this system only has a few components, it will only need a few packages/libraries in order to implement. First off, it will need WordNet [2] in order to find the meanings of and synonyms for the various words. It will also need the NLTK toolkit [1] in order to tag phrases correctly. Another useful API might be the Wolfram|Alpha API [3], which can be used to improve part 2 of the problem when the system extracts the meanings from each phrase. However, the Wolfram|Alpha API is not entirely necessary, and the task can still be completed using NLTK and WordNet. The added API might help to improve accuracy and precision, but there would also need to be added sections on how to interact with the API, and how to convert between Wolfram|Alpha and WordNet / NLTK meanings.

The only other component that would need to be added is the clustering algorithm, which is fairly simple to implement from scratch. It will also be useful to implement from scratch, because it can be tailored to the exact needs of the system, and produce the correct features in the domain we have specified. We can also experiment to see if it helps to add each calculated phrase to the feature matrix, or if we should not 'learn' any of the data we are currently parsing.

6. Evaluation

As mentioned in the problem statement, this system can be used in a larger program that attempts to play the 'optimum' card for a given judge, black card, and hand of white cards. Because of this, the system will be evaluated using the extrinsic evaluation of how well the AI performs using various tagging techniques, i.e. what percentage of rounds it plays the winning card for a given judge. As a baseline, this NLP system can be run against the AI my CS 4701 partner and I made that uses hand-tagged features for each of the cards. The machine learning algorithms and everything else for the AI would be the exact same, the only difference being the method by which the cards were tagged with feature vectors. We can also experiment with various versions of the NLP system, such as adding tagged phrases to the feature matrix for the clustering algorithm, using Wolfram|Alpha API or not, etc. and see which yields the best performance rating.

In order to effectively evaluate the performance of the system, the AI will need to be tested against multiple judges with various preferences. In order to do so, you can either use a number of different students, or bias yourself by always selecting answers that fall into one feature category such as Sex or Religion. For each new judge, you need time and example for the machine learning aspect learn their preferences, then once the AI has sufficiently learned their preferences (use the same number of training examples for each judge), you can record the performance of the NLP system. This method worked well for our CS 4701 project, and as long as everything else is held constant, the same method can be used to test the effectiveness of the various NLP systems and versions.

7. Anything else...

I should probably mention that the original version of this game is called *Apples to Apples* and is much more family friendly. Instead of white and black cards each containing phrases, it has 'red cards' and 'green cards', containing nouns and adjectives, respectively. Apples to Apples is a family-friendly game, and can be played by young children. Cards Against Humanity, however, is rated for ages 17^{+}, and the subtitle on the box reads "A party game for horrible people". The NLP system described above should work just as well for Apples to Apples as it does for Cards Against Humanity, but the features and some of the processing steps would need to be altered in order to fit the different format. It could also simply solve the problem "return the red card from the hand that the green card best describes", and would perform rather well with a naive audience.

The reason why I chose the version of the game I did was because my CS 4701 partner and I had already implemented a machine learning system for playing the ‘optimal’ card for a given judge, and the NLP task would be an interesting extension to add. Also, I have a copy of the Cards Against Humanity game in my dorm room so I was able to look at examples of cards to help describe my system, whereas I would have had to buy a copy of Apples to Apples in order to use those as examples. Plus, Cards Against Humanity makes for a *much* more interesting NLP problem compared to simply pairing nouns with adjectives, as the phrases can be much longer and more convoluted.

8. Individual Member Contribution

I did 100% of the work on this project. I had a bad experience working with a partner on previous project in this class, so I decided to work by myself for this assignment. I also find working by myself a lot easier, because I can work on it whenever I have free time, rather than trekking all the way to central campus and meeting with a partner for an hour or so, then going home and working on my section of the assignment.

References

[1] Bird, Steven, Ewan Klein, and Edward Loper (2009), Natural Language Processing with Python, O’Reilly Media.

[2] Princeton University “About WordNet.” WordNet. Princeton University. 2010. http://wordnet.princeton.edu.

[3] “Wolfram|Alpha API Overview Documentation Language Libraries API Explorer FAQs Terms of Use Sign In.” Wolfram|Alpha API: Access Data and Calculations for Your Applications. Wolfram|Alpha, n.d. Web. 03 Dec. 2015. http://products.wolframalpha.com/api/.