Week 10, continued

This is CS50. Harvard University. Fall 2015.

Anna Whitney

Table of Contents

1. Introduction

- We are excited to invite two of our friends from Yale up to talk about artificial intelligence, language processing and more!
- Our head TF from Yale, Andi, introduces Scaz.
- Scaz runs the robotics lab at Yale, and brought one of the robots with him to show us.
- There will be the CS50 Research Expo at Yale on Friday, November 20, from 3-4:30 PM in the Sterling Memorial Library. All CS50 students & staff are invited, as well as researchers, student groups, and startups from Yale and Harvard related to engineering and science.
  - # If you're coming down for the Harvard-Yale game and getting there a little early, you should definitely check it out!

2. Natural Language Processing

- Natural language processing (NLP) is a way of thinking about a new way to communicate with our devices.
- We distinguish between computer languages and "natural languages", which humans use to communicate with other humans, and NLP tries to bridge the two.

- Probably the first NLP system was **ELIZA**, designed by Joseph Weigenbaum in 1966 to mimic Rogerian psychotherapy, where the therapist mirrors back the feelings of the patient.
- Volunteer Miles comes up on stage to demonstrate what ELIZA can do:

```
HI! I'M ELIZA. WHAT'S YOUR PROBLEM?
I can only fall asleep with bunny slippers on.
SAY, DO YOU HAVE ANY PSYCHOLOGICAL PROBLEMS?
Yes.
YOU SEEM QUITE POSITIVE.
No, in fact, I am an electron.
DID YOU COME TO ME BECAUSE YOU ARE AN ELECTRON?
```

  # ELIZA clearly responds to some pieces of what you're saying, but without any real understanding.
  # This is a kind of system called **pattern matching**, where we're taking bits of text that were provided by the user, performing some transformation on them, and returning them back to the user.

- ELIZA takes a sentence like `I want to impress my boss.` and looks for certain patterns. One of ELIZA's patterns is `I want`, and any time it sees that pattern, it formulates a response starting with a fixed string, in this case `WHY DO YOU WANT*`.
  # The asterisk at the end indicates that this is just going to be the beginning of ELIZA's response, and the rest of the user's input will be filled in to finish the sentence:

```
WHY DO YOU WANT*TO IMPRESS MY BOSS.
```

  # But there's actually a little more processing that goes on here, to convert pronouns, etc.
  # Rather than just sticking the rest of the user's utterance on the end, we'll take the words - the string tokens - one at a time and perform conversions as necessary:

```
I want to impress my boss.
-
WHY DO YOU WANT
WHY DO YOU WANT TO
WHY DO YOU WANT TO IMPRESS
WHY DO YOU WANT TO IMPRESS *YOUR*
WHY DO YOU WANT TO IMPRESS YOUR BOSS
```

```
WHY DO YOU WANT TO IMPRESS YOUR BOSS_*?*_
```

- This is a very simple system, but it worked well enough that when Weigenbaum set it up on a machine in his lab in 1966, he eventually had to restrict access to it because his students were chatting with it rather than getting any work done.

  # In fact, he had to fire his assistant, who spent all her time talking to ELIZA about her problems.

  # Everyone who talked to ELIZA anthropomorphized it, to the point that psychotherapists worried they would be replaced.

- Let's look at how this was implemented (the full source code is on the course website[1])[2]:

  # First, we have a list of keywords:

```
char *keywords[]= {
        "CAN YOU","CAN I","YOU ARE","YOURE","I DONT","I FEEL",
        "WHY DONT YOU","WHY CANT I","ARE YOU","I CANT","I AM","IM ",
        "YOU ","I WANT","WHAT","HOW","WHO","WHERE",
        "WHEN","WHY",
        "NAME","CAUSE","SORRY","DREAM","HELLO","HI ","MAYBE",
        " NO","YOUR","ALWAYS","THINK","ALIKE","YES","FRIEND",
        "COMPUTER","CAR","NOKEYFOUND"};
```

  # These are things that ELIZA will recognize and respond to directly.

  # We also have a set of words to swap, like converting `MY` to `YOUR`:

[1] http://cdn.cs50.net/2015/fall/lectures/10/w/src10w/eliza.c.src

[2] This isn't particularly well-written code, and there are definitely optimizations that we'd expect you to be able to make, but we tried to keep it as true to the original as possible.

```
char *SWAPS[NUMSWAPS][2] = {
    {"ARE","AM"},
    {"WERE", "WAS"},
    {"YOU","I"},
    {"YOUR", "MY"},
    {"IVE", "YOU'VE"},
    {"IM", "YOU'RE"},
    {"YOU", "ME"},
    {"ME", "YOU"},
    {"AM","ARE"},
    {"WAS", "WERE"},
    {"I","YOU"},
    {"MY", "YOUR"},
    {"YOUVE", "I'VE"},
    {"YOURE", "I'M"}
};
```

And then we have prewritten responses, with multiple responses for each scenario, so that if we say "yes" to ELIZA four times in a row, we could get four different responses:

```
char *responses[NUMKEYWORDS][9] = {
    {   "DON'T YOU BELIEVE THAT I CAN*",
        "PERHAPS YOU WOULD LIKE TO BE ABLE TO*",
        "YOU WANT ME TO BE ABLE TO*"},
    {   "PERHAPS YOU DON'T WANT TO*",
        "DO YOU WANT TO BE ABLE TO*"},
    ...
    {   "YOU SEEM QUITE POSITIVE.",
        "ARE YOU SURE?",
        "I SEE.",
        "I UNDERSTAND."},
    ...
    {   "SAY, DO YOU HAVE ANY PSYCHOLOGICAL PROBLEMS?",
        "WHAT DOES THAT SUGGEST TO YOU?",
        "I SEE.",
        "I'M NOT SURE I UNDERSTAND YOU FULLY.",
        "COME, COME ELUCIDATE YOUR THOUGHTS.",
        "CAN YOU ELABORATE ON THAT?",
        "THAT IS QUITE INTERESTING."}
};
```

In the actual `main` function, we initialize some variables, do some housekeeping, but then we get to the very understandable loop that actually controls ELIZA's behavior:

```
int main(int argc, char * argv[])
{
    ...
    while (1) {
        readline(inputstr);

        // check for termination
        if (strcmp(inputstr,"BYE")==0)
            break;

        // check for repeated entries
        if (strcmp(lastinput,inputstr)==0)
        {
            printf( "PLEASE DON'T REPEAT YOURSELF!\n");
            continue;
        }
        strncpy(lastinput,inputstr,strlen(inputstr)+1);
...
```

First, we check for the special termination keyword `BYE`, and make sure the user isn't just repeating themself.

```
...
        // see if any of the keywords is contained in the input
        // if not, we use the last element of keywords as our default
 responses
        char *location;
        strcpy(reply,"");
        for(k=0;k<NUMKEYWORDS-1;k++)
        {
            location=strstr(inputstr, keywords[k]);
            if(location != NULL)
                break;
        }
...
```

```
# ELIZA then looks for keywords, and if there aren't any, uses a default response
  that tries to be broadly applicable.
```

```
...
        // Build Eliza's response
        // start with Eliza's canned response, based on the keyword
 match
        baseResponse = responses[k][whichReply[k]];
        baseLength = strlen(baseResponse);

        if(baseResponse[baseLength-1] != '*')
        {
            // if we have a baseResponse without an asterix, just use
 it as-is
            strcat(reply, baseResponse);
        }
        else
        {
            // if we do have an asterix, fill in the remaining with
 the user input
            // use all but the last character of the base response
            strncat(reply, baseResponse, baseLength-1);

            // now add in the rest of the user's input, starting at
 <location>
            // but skip over the keyword itself
            location+=strlen(keywords[k]);
            // take them one word at a time, so that we can
            // substitute pronouns
            token = strtok(location, separator);
            while(token != NULL)
            {
                for(int s=0;s<NUMSWAPS;s++)
                {
                    if(strcmp(SWAPS[s][0],token) == 0)
                    {
                        token=SWAPS[s][1];
                    break;
                    }
                }
                strcat(reply, "");
                strcat(reply, token);
                token=strtok(NULL, separator);
            };
            strcat(reply, "?");
        }
        printf("%s\n", reply);
...
```

    # We then construct ELIZA's response the way we talked about before.

- ELIZA had a huge impact on natural language processing because it made it seem like NLP was very attainable.

3. Speech Recognition

- Our first step to go beyond ELIZA is to shift away from dealing with text typed in to a prompt, and instead be able to understand spoken text.
- We'll have to build a set of models to turn the low-level acoustic information into words and sentences.
    # We follow a statistical model that starts with phonetics, to identify what letters correspond to the sounds.
    # On top of that, we'll build a word pronunciation model that combines these into a word.
    # We then need a language model to combine words into sentences.
    # All of these models are really just going to be statistics, so we can work with them all simultaneously.
- A **phonetic model** is based on a **Hidden Markov Model**, where we keep track of a "state of the world", and interpret that as part of an action. We have transitions that let us move from one state to the next with some probability.
    # For example, when you make an M sound, there's a short intake of breath, a sort of humming with your lips together, and then a plosive, or expelling of breath.
    # The model is designed to take into account that different people may have slightly different timing of these stages, or that the same letter comes out slightly differently in different words.
    # To recognize a particular sound, we build a hidden Markov model of every letter we have, and then determine which has the highest probability of creating the sound.
- We then combine our letter sounds into a **word pronunciation model**, also based on a hidden Markov model, although this can be different from the phonetic model because one word could have diverging pronunciations (like to-may-to vs. to-mah-to).

    # We need multiple paths because any speech recognition system is going to have to work with lots of differen speakers with different accents and even different uses of the same words.

- On top of that, we have a complicated-looking **language model**, but it's actually the simplest of the three.

    # The language model operates on what's called an **N-gram model**, which determines how likely a given word is to appear after the previous word or words.

    # For example, a **bigram model** determines its probability on just the immediate previous word (so the probability of the sentence `The rat ate cheese.` would be the probability of a sentence starting with `The`, times the probability of the word `rat` following the word `the`, times the probability of the word `ate` following the word `rat`, times the probability of the word `cheese` following the word `ate`).

    # Similarly, with a trigram model, we could indicate that the words `Weather forecast` are likely to be followed by the word `today` or `tomorrow` but unlikely to be followed by the word `artichoke`.

    # We can build an N-gram language model by counting the occurrences of each N-gram in a large corpus of text (e.g., every Wall Street Journal published since 1930).

    # If we do this with a large enough sample of data, it works really well.

- Most operating systems come with voice recognition at this point, and these things are based upon this sort of three-layer model.

    # They have to do a little more than this to actually answer questions, but these three steps are what it takes to recognize what you're saying.

- If we ask Siri a question like "Siri, what is the weather like in New Haven today?", we can first see that Siri recognizes each of the individual words, and then produces a response.

    # Now that we know these models are just statistical, we can play some games with Siri, like asking "Siri, what is the weather hippopotamus New Haven today?" - and we'll see that even though Siri does recognize the word `hippopotamus` in our question, it gets ignored and the exact same weather results are returned, because Siri has figured out the pattern.

    # However, if we go far enough away from this model - like asking "Siri, what is the weather armadillo artichoke hippopotamus New Haven?" - Siri can't pick up on the

`weather` + `New Haven` pattern and instead resorts to just giving us search results instead.

4. Question Answering

- If we want to think about how these systems actually respond to what we say, we need to talk about the more fundamental topic of **question answering**.
- In the same way that our speech recognition model is composed of three layers, to actually "understand" and respond to something the user says, we again rely on a multi-layer analysis of the text that's being analyzed.
- The first component is often **syntactic processing**, trying to understand the structure of a sentence.
  - # This includes recognizing which words are nouns, verbs, etc, as well as determining whether a given sentence structure is grammatically valid within a particular grammar (e.g., English grammar).
  - # This helps, but isn't enough on its own to determine the meaning of a sentence.
- Next, we need some amount of **semantic processing** - what is the meaning that each word carries?
  - # We associate with each word a function, a certain transformation that it allows to happen.
    - # For example, in the sentence `John loves Mary.`, `John` and `Mary` are proper names that confer identity, while `loves` implies a particular relation between its arguments (`John` and `Mary`) that we can label and manipulate (note that this doesn't mean the model understands what love is, but it does understand how it fits into a sentence and what other parts of the sentence it relates to).
  - # Any type of semantic processing requires a little bit of knowledge and a lot of work on our part - just plain statistics aren't generally enough to generate a semantic model.
  - # Most of these systems require humans to spend a great deal of time encoding how these sentences can be represented semantically.
- It gets more complicated than that - even once we've dealt with semantics, we have to understand the **pragmatic** content of what's being said - i.e., how do I relate the

words in the sentence to something real in the world, or at least to some information source out there?

  # Sometimes there are ambiguities that semantics can't resolve, e.g.:

```
Red-hot star to wed astronomer.
Squad helps dog bite victim.
Helicopter powered by human flies.
```

    # For all of these sentences, there are multiple equally grammatically valid interpretations, but they don't make sense equally.

  # Syntax and semantics can't distinguish between these interpretations! But question-answering systems need to be able to tell the difference.

  # There have been huge improvements in these systems in the last decade.

  # IBM's Watson is one of the best examples of a question-answering system today.

- These systems have to rely upon recognizing the speech, turning it into a meaningful internal representation, and then going out and finding an information source to answer the question.

  # These methods aren't much different from what you've learned in this class! We can parse HTTP requests, get information from an external database, and so on.

- Even Watson can't answer an arbitrary question about any topic. Questions have to be formatted in certain ways and in certain domains.

  # There are various domain-specific question-answering systems in areas like medical informatics and even food-beer pairings.

  # However, they don't work across domains - asking the food-beer system about medical issues won't work.

  # We're still hand-coding the infrastructure to make these run.

5. Non-verbal Communication

- A great mass of information that we communicate with each other doesn't come about through the individual words that we say. We also use gaze, tone of voice, inflection, etc.

    # Siri doesn’t care about these differences, giving the same answer regardless of tone of voice.
    # Some interfaces, though, do react to those differences, such as a robot that Scaz brought with him named Jibo.
        # Jibo is built to be interactive, and not only via speech - it also engages in non-verbal ways, moving around to acknowledge positive and negative tones.
        # Volunteers Alfredo and Rachel demonstrate complimenting and insulting Jibo, and Jibo very clearly has different "body language" in response to the two.
        # We then get a demo of Jibo dancing.

- These types of communication, though non-verbal, have similar rules to linguistic communication and similar purposes - to have an effect on the person communicating with the system.
- Next week, we’ll talk about how to interface with computer opponents in games.