Jabberwocky: You don't have to be a rocket scientist to change slides for a hydrogen combustion lecture

David Franklin, Shannon Bradshaw and **Kristian Hammond**
Intelligent Information Laboratory
Northwestern University
{franklin, bradshaw, hammond}@infolab.nwu.edu

Abstract

In designing Jabberwocky–a speech-based interface to Microsoft PowerPoint–we have tried to go beyond simple commands like "Next slide, please" and make a tool that aids speakers as they present and even learns as they rehearse their presentations. Jabberwocky looks at the contents of the slides, extracting key words and phrases and associating them with their places in the presentation. By listening for these phrases (and synonymous phrases derived using syntactic rules) Jabberwocky is able to follow along with the presentation, switching slides at the appropriate moments. In this paper, we discuss the implementation of this systema component of our Intelligent Classroom project–and look at how we are using it.

1 Introduction

In the Intelligent Classroom, we are enabling new modes of user interaction through the use of multiple sensing modes and plan recognition–the Classroom uses cameras and microphones to determine what the speaker is trying to do and then takes the actions it deems appropriate. It controls automated VCRs and slide projectors and also produces a video of the presentation. One of our goals is for the speaker to be able to interact with the Classroom as he would with an A/V technician: sometimes through commands (speech and/or gesture), and often by just going about his presentation and trusting the Classroom to do the right thing.

The Jabberwocky system is the result of our incorporation of Microsoft's PowerPoint presentation software into the Intelligent Classroom. In addition to some simple command-based approaches, we have implemented two techniques based on slide contents. In the first, the system is listening for particular words and phrases that indicate when the speaker is going on to a new slide or point. The second utilizes some probabilistic techniques to match the speaker's words to the content of slides; the matches indicate which slide to show. By combining all of these approaches, Jabberwocky is able to support a wide range of presentation styles, allowing speakers to lecture in whichever manner they feel will be the most effective. For example, the probabilistic approach lets the speaker skip around in his presentation by describing the slide that he wishes to skip to–this would prove invaluable for answering audience questions. In the extreme, this could even support a speaker who wished to do an unstructured lecture from a large body of slides.

In this paper, we focus on the speech-based aspects of Jabberwocky. (The computer vision and plan recognition aspects are detailed in other Intelligent Classroom papers (Franklin 1998) (Franklin & Flachsbart 1998).) First we look at the ways people tend to lecture and at how Jabberwocky should be implemented to support this. Then we look at the basic operation of Jabberwocky: listening for particular words, phrases and commands that indicate when the speaker wishes to change slides. Then we discuss how Jabberwocky extracts key words and phrases from the slide contents and how these can be used to follow along with a strictly linear slide presentation. Then we look at a probabilistic approach to determining what slide the speaker is lecturing from, based on what words he uses. Then we present two examples of Jabberwocky at work. Finally, we summarize this work and look at some future directions for the research.

2 How do people lecture?

Our primary goal as we implemented Jabberwocky was to construct something that we would actually use. To address this we needed to envision how such a system would be used and then make sure that the system's requirements were reasonable to attain. We did not

wish to invent new ways of lecturing but instead to enable people to utilize slides in their preferred lecture styles.

2.1 Lecture styles

One curse of overhead transparencies, slides and PowerPoint is that the speaker is essentially forced to give his presentation in precisely the same way every time. To deviate from the prescribed order, he must either rifle through his set of overhead transparencies, skip over all the slides (one at a time) between the current slide and the desired one, or, in PowerPoint, find the slide in a buried menu. As a result of this inconvenience, speakers tend to let the slides dictate the presentation instead of support it.

There are, of course, speakers who refuse to be deterred. A speaker from our laboratory once gave a presentation using a box of more than one hundred overhead transparencies. He had painstakingly organized them into at least twenty categories and did numerous practice talks to become proficient at extracting slides to respond to particular questions. During his conference presentation, an audience member asked one of the questions he had a slide prepared for, and he quickly located the appropriate slide and displayed it. As the audience murmured its admiration, he said "I'm glad you asked that!" However, many speakers are not willing to go to such lengths to prepare for a presentation–nor should they have to be.

With PowerPoint, speakers may use hypertext links within their slides to create flexible presentations. However, this requires a great deal of preparatory effort and also forces the speaker to use the mouse for much of his navigation through his lecture. The point here is that although it is possible to give flexible presentations using slide-based media, almost no one even tries to, dissuaded by the effort.

Perhaps it is not surprising that many of the most dynamic and spontaneous teachers avoid using slides in their presentations. They wish to address students' questions or reorganize their lecture on the fly to meet their particular students' needs. They wish to explore interesting tangents when the lecture provides an opportunity. It is very difficult to do these sorts of things in the confines of a traditional slide-based lecture.

2.2 Supportive technologies

In our design, we wish to make use of existing technologies as much as possible; we use IBM's ViaVoice software for voice recognition and Xerox's part of speech tagger as a part of our syntactic analysis of the slide contents.

ViaVoice supports two modes of voice recognition: continuous dictation and command-based. In the continuous dictation mode, ViaVoice attempts to recognize every word that is said, using the surrounding words as clues to help with ambiguous words. The software promises 95% accuracy, which can be achieved when the system is trained and the speaker is consciously over-enunciating. When a speaker talks conversationally, the accuracy can easily fall below 70%. In the command-based mode, ViaVoice listens for specific commands. The command language is specified using a context-free grammar. The software expects commands to be preceded and followed by brief pauses–this is how commands are distinguished from dictation. The accuracy of the command-based mode seems very good.

Xerox's part of speech tagger (Xerox) uses a hidden Markov model to find the most probable syntactic parse of a sentence or phrase. It utilizes a lexicon of nearly 60,000 words, including common proper names and informal language.

3 The basic operation of jabberwocky

Basically, Jabberwocky uses the speaker's slides as a rough outline. As the speaker goes about his presentation, Jabberwocky listens for clues as to where the speaker is in this outline. Usually, the clues confirm that Jabberwocky understands where they are, but occasionally, the clues will indicate that Jabberwocky is lost. (A speaker who tends to go off on tangents or who just skips around a lot is likely to confuse the system–and his listeners.) These clues take two forms: (1) words and phrases that are indicative of particular slides and slide points and (2) commands that indicate how the speaker wishes to move through his presentation.

Before the speaker begins his presentation, Jabberwocky analyzes the slides and constructs a hierarchical "plan" for following along with the presentation. The steps in the top-level plan correspond to individual slides, and each slide's plan has steps corresponding to its key points. These steps are essentially of the form: "if you hear the speaker say 'probability distribution' then he must be discussing the third point." If it is able to choose these "clues" well, Jabberwocky will have enough information to keep track of the speaker's place during the presentation.

Once the presentation has begun, Jabberwocky listens for the set of clues appropriate to its current place in the presentation. It will listen for the key words and

phrases associated with the current slide (to keep track of which point the speaker is on) and the next slide (to see when the speaker has gone on), and for the various commands that are appropriate to the slide presentation. As the speaker moves through the presentation, Jabberwocky changes which words, phrases and commands it is listening for.

Commands allow speakers direct control of their presentation. This level of control may be needed to remedy system mistakes (like switching slides too early or too late). Also, this level of control may be useful when indirect control is awkward or infeasible. For instance, while a well-rehearsed speaker may have no difficulty leading Jabberwocky, a less confident speaker may wish to allow Jabberwocky to lead him. Such a speaker often does not know what is on the next slide, and needs to command "next slide" to find out. In addition, almost any speaker would prefer saying "skip ahead to the conclusion slide" to trying to describe its contents. The various commands used in Jabberwocky will be discussed as appropriate through out the paper.

4 Choosing important words and phrases

To be successful, Jabberwocky must do a good job of extracting key phrases from the slidesthese phrases serve as Jabberwocky's "understanding" of the presentation. This is in lieu of a rigorous natural language understanding of the presentation, which is simply not feasible for Jabberwocky. Fortunately, this deeper understanding is not necessary for Jabberwocky; a human A/V assistant can match what the speaker is saying to the contents of the slides without any technical knowledge of the presentation's subject matter. It appears that a purely syntactic understanding is sufficient for knowing what to do. It is like Alice's situation after reading the poem Jabberwocky in Lewis Carroll's "Through the Looking-Glass." (Carroll 1871) The poem was nonsense to her, but she was still able to glean a little meaning from it: "Somehow it seems to fill my head with ideas–only I don't exactly know what they are! However, somebody killed something: that's clear at any rate."

4.1 Extracting key phrases from the slides

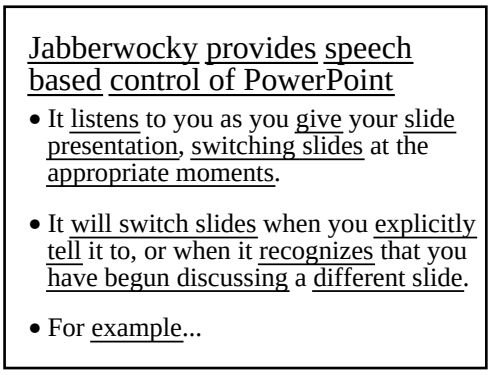
The primary way that Jabberwocky locates words and phrases that are representative of slides and slide points is through syntactic analysis. Much of the meaning of a sentence can be grasped just by looking at the various actors and actions in the sentence. In discussing a slide point, the speaker can be expected to mention its actors and actions. They may be discussed in a different order than they appear in the slide (i.e. changing a sentence from the passive to the active voice). The speaker may even dramatically change how they are described (i.e. converting adjectives into prepositional phrases). These potential problems are dealt with by choosing phrases to independently represent the actors and actions and by creating numerous synonymous phrases for actors and actions that can be expressed in many ways.

Jabberwocky locates phrases for the important actors and actions in a sentence by first finding all the noun and verb phrases, and filtering out words and phrases that are not "interesting." Articles, pronouns, propositions and words that essentially serve as syntactic glue are filtered out. This filtering process often eliminates uninteresting phrases from consideration and also can strip a cumbersome phrase down to its key elements. Jabberwocky considers these key elements (and the paraphrases developed using them) as representative phrases for the sentence. (These techniques for selecting important words and phrases have also been employed on Rosetta (Bradshaw & Hammond 1999), a system that indexes research papers based on how they have been cited, and on DRAMA (Wilson & Bradshaw 1999), which uses free-form text in its indices for case-based retrieval.)

Jabberwocky also looks at non-text objects in the slide when creating key phrases. With slides containing graphs, tables or pictures, we can anticipate that the speaker will mention them in discussing the slide. So, for a slide with a table, Jabberwocky will also listen for phrases like "in this table" and "in the table on the right." For each of the different objects we expect to find embedded in a PowerPoint slide, we have enumerated a number of key phrases that could refer to it.

4.2 How do people paraphrase their slides?

When designing slides, speakers tend to be as concise as possible–putting as much content as possible into just a few words. As a result, slides are often full of convoluted language; overly complicated noun and verb phrases abound. But, when speaking, people tend to use more natural language, avoiding the stilted prose of their slides. Therefore, it is unreasonable to assume that speakers will describe actors and actions (in the slide contents) using exactly the same words as the slides do. So, for each actor or action phrase,



jabberwocky, provide, speech, base, control powerpoint, control, powerpoint

listen, give, present slide, slide present, present, slide switch, switch slide, present, slide switch, switch slide, slide, appropriate moment, moment, appropriate

slide switch, switch slide, switch will, will switch, switch, will, explicitly tell, tell, recognize, begun discuss, discuss, have begun, different slide, slide

example

Figure 1: Phrase extraction in Jabberwocky. On the left, a slide, with identified phrases underlined. On the right, the words and phrases that Jabberwocky will be listening for.

we construct a number of synonymous phrases, using a number of syntactic transformation rules.

With verb phrases, we construct a phrase for each way the adverbs and helping verbs can be moved or eliminated. From the verb phrase "can immediately advance", we also get "can advance immediately" and "immediately advance" (among others). Also, since our matcher converts all verbs to their root forms, the phrases also account for verb conjugation changes due to tense, plurality or use of the passive voice.

With noun phrases, we break the phrase into smaller units: the adjectives, the modifier nouns and the subject noun. From the noun phrase "current information management technology" we get "current", "information management" and "technology" for the three units. Given these, we construct key phrases using the following rules:

- The modifier nouns can serve as a key phrase (as in "information management").
- The modifier nouns can be separated from the adjective and subject noun (as in "current technology for information management").
- Different subsets of the adjectives and modifier nouns can be used (as in "information technology").

Finally, we take one and two word subsequences of the resulting phrases–this makes the system more tolerant of words being missed by the speech recognizer. Using these rules for converting noun and verb phrases into a large number of words and phrases, Jabberwocky is able to cover most of the ways a speaker will paraphrase their slides, without diluting their meaning so much that it makes wishful false matches.

In Figure 1, we see the verb and noun phrases that Jabberwocky identified in an example slide (on the left) and the words and phrases that Jabberwocky derived from them (on the right). It is important to note that the phrases on the right are made up of words in their root forms and also that they omit certain connecting words like prepositions and possessives. So the phrase "present slide" would be found in the phrases "when you are presenting slides" and "in presenting your slides" among others.

5 Skipping around in the slides

During the course of a typical presentation there are often situations where the speaker needs to deviate from the planned presentation: an audience member's question addresses a previous slide; time constraints require the speaker to condense his presentation. Sometimes the speaker may not even have a planned presentation; he may have a body of slides that he wishes to use during an improvised discussion. In this section, we look at how the word and phrase "clues" from the previous section, coupled with a probabilistic approach, can support these sorts of dynamic presentations.

5.1 A probabilistic foundation

Our probabilistic approach utilizes Bayes' Law to update a probability distribution across the set of slides. When Jabberwocky decides that the speaker may want to skip to another slide, it will first assign an initial probability to each slide (based on how likely it is that the speaker will skip to it). Then, after each word or phrase that it hears, it will update these probabilities

in response (using Bayes' Law). Finally, when one slide clearly dominates, Jabberwocky switches to that slide.

$$P(s|w) = \frac{P(s)P(w|s)}{P(w)} \quad (1)$$

The s's refer to the indices of slides and the p's refer to the key words and phrases that have been extracted from the slide content. In addition:

- $P(s|p)$ is the probability that the speaker wants slide s given that he just said phrase p. We compute this using the other values.
- $P(s)$ is the probability (immediately prior to hearing phrase p) that the speaker wants slide s. $P(s)$ is simply the last value of $P(s|p')$ (where p' is the most recent phrase). Jabberwocky uses $P(s)$ as an intermediate value; $P(s|p1, p2)$ is equal to $P(s)$ after hearing phrases $p1$ and $p2$ (assuming that the probabilities are independent.)
- $P(p|s)$ is the probability that the speaker would say phrase p, if he wants slide s. If the speaker were to simply read verbatim from the slides, $P(p|s)$ would simply be the number of times phrase p appears in slide s divided by the number of phrases we have extracted from the slide. Since most speakers do not just read the slides, we also consider it possible (though less likely) that the speaker will use phrases that appear elsewhere in the presentation. (For most presentations there will be a number of important phrases that are spoken repeatedly throughout the presentation, but that will only appear in a few slides.) So, we compute $P(p|s)$ as a weighted sum of the probabilities based on phrases in the slide and phrases in the presentation.
- $P(p)$ is the probability that the speaker would say phrase p in the current situation. This is simply the sum of all the $P(p|s)$, weighted by $P(s)$.

5.2 Fitting it into Jabberwocky

In a situation where the speaker wants to skip to a particular slide, he will tell Jabberwocky (through a command) that he wishes to skip and then will begin discussing the desired slide. For example, the speaker might say "Please skip back ¡pause¿ to the slide that talks about how Jabberwocky extracts phrases from the slide contents." By the time Jabberwocky has updated the probabilities for the phrases "extracts", "phrases" and "slide contents" the probability for the desired slide will be sufficiently high that it is switched to.

When the speaker makes a slide-skipping command, Jabberwocky must first set up the probability distribution by assigning initial values to the $P(s)$. Based on the particular command, Jabberwocky may be able to reduce the set of slides to consider. For example, if he says "skip back..." then we need only consider slides that have been seen already. Jabberwocky distributes the probability evenly among the slides that it is considering.

Then, as the speaker talks, Jabberwocky listens for all the phrases it has extracted from the set of slides under consideration. When it hears one of these phrases, it computes $P(s|p)$ for each slide and updates the probability distribution. Then if the probability for one slide is sufficiently high (90%), it will skip to that slide and continue the presentation from there. Otherwise, Jabberwocky will continue listening for phrases and updating the probabilities.

In a truly freeform lecture (where the speaker is treating his slides as a set–rather than sequence), Jabberwocky will remain in the probabilistic mode indefinitely. In this case, the probability distribution is set up to favor slides that have not yet been viewed. But also, when updated, the probabilities are adjusted such that no slide becomes too improbable. We will need to do more practical experimentation to determine how to best constrain the probabilities and facilitate this mode of lecturing.

6 Learning what the speaker might say

There are two situations in which Jabberwocky fails to match the speaker's words to the correct slide: when the voice recognition drastically mis-hears what the speaker said and when the speaker uses too many synonyms in paraphrasing a point. In both cases, Jabberwocky's shallow understanding (of phonetic similarity and of synonymy) renders it incapable of making the match.

When we examined the first situation, we immediately noticed that the mis-hearings were somewhat consistent. With a phrase like "information systems", if the speech recognizer heard "and permission systems" once, it was likely to hear it that way much of the time. This means that if Jabberwocky listens while the speaker practices his presentation (even just a few times), it will encounter most of the recognition errors for that presentation. Better still, this makes it unnecessary for the speaker to go through the tedious process of training the voice recognition system.

With the second situation, we were faced with a frus-

The speaker said: You might wonder: "How does it know what I'm talking about?"

Jabberwocky heard (from the speech recognizer):	Phrases that were learned:
1) "You might wonder how does annoy them talking about"	do annoy, annoy, how do, wonder how, wonder might, might wonder, wonder, talk
2) "You wonder how does the northern talking about"	how do, wonder how, wonder, northern talk, talk
3) "You wonder how is it now been talking of a"	wonder how, wonder, now talk, talk

Figure 2: Learning phrases in Jabberwocky. Across the top, what the speaker said. On the left, what Jabberwocky heard the three times the speaker spoke. On the right, the words and phrases that Jabberwocky learned.

trating problem. Though we would like to use something like WordNet (WordNet) to capture semantic similarities between words, in technical presentations, it will not find the important synonyms that are specific to a particular technical field. So, pending a better solution, we simply memorize phrases that seem potentially synonymous, so that they can be recognized later.

6.1 What words should be remembered?

In our methods for dealing with the situations where the basic techniques fail, we have consciously avoided forcing Jabberwocky to rely on a deep understanding. Instead, we rely on the speech recognizer and speaker to be reasonably consistent so that we can simply match their words and phrases against their past words and phrases. To determine which words and phrases are worth remembering, Jabberwocky uses the same phrase extraction methods that it uses in analyzing the slide contents. In addition, because ViaVoice's vocabulary is much larger than the lexicon in Xerox's part of speech tagger, we also extract large words that are not successfully tagged. This deals successfully with both mis-hearings and with obscure technical words.

6.2 How does the learning take place?

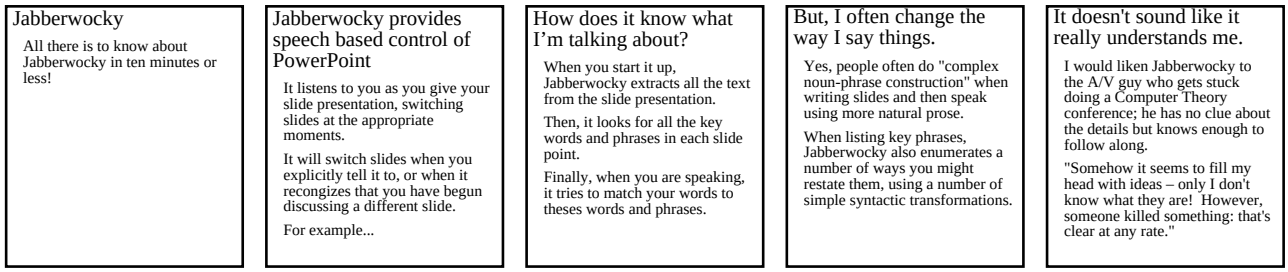
Jabberwocky must do more than just identify what words and phrases are important to remember; it must also know what slides or slide points to associate them with. (If it makes many bad associations, Jabberwocky is essentially learning how to do a bad job of running the slide presentation.) Jabberwocky simply associates the phrases with the slides (and even slide points) that they were heard with. Because of this, while rehearsing with Jabberwocky, the speaker should make sure that they stay well synchronized.

When they do get out of synch, the speaker can inform Jabberwocky by issuing commands like "next slide" or "no, go back." The phrases immediately preceding the command are then associated either with the next slide (for the first command) or with the previous slide (for the second). A speaker may further aid Jabberwocky by telling which slide points he is lecturing from (as in "here is slide point one.") With speakers who reliably discuss every point in order on their slides, this level of training should help Jabberwocky to provide error-free control of their slides.

In Figure 2, we examine how Jabberwocky learns from the narration it hears. On the left, we see the text from a slide title and three ways that Jabberwocky heard the speaker discuss it. The speaker added the phrase "you might wonder" as a transition into the slide and Jabberwocky learned several phrases involving the word "wonder." Now Jabberwocky can anticipate the speaker's transition. The speech recognizer had problems with the phrase "does it know what I'm" (monosyllabic words tend to cause the most difficulty). The phrases it learned as a result (involving "annoy" and "northern") will help it the next time the speech recognizer mis-hears the words in the same way.

7 Experiments

In this section we take a look at Jabberwocky in action. First, we do a free-form lecture on a set of five slides, showing how the words and phrases extracted from the slides allow Jabberwocky to identify which slide the speaker is lecturing from. Second, we train Jabberwocky by rehearsing a presentation three times and see the words and phrases that it learns. Jabberwocky is implemented on a network of a Macintosh (for text processing using Common Lisp) and a PC (run-



<pause> today it all to you ever thing to one know about jabberwocky 10 minutes to less <pause> so for stop what does into <pause> the essentially provides a **speech** based interface to microsoft's power plant presentation software <pause> it listens is to presenters lads switching sides of the appropriate moments <pause> it switches the slide when it's possibly tell until when recognizes the jurors discussed in different <pause> for example <pause> new wonder how is the norm talking about <pause> while winning **started** up jabberwocky to extract the text premise slides in the presentation <pause> then defines key words and phrases and in each from <pause> finely when you speak it matches your words to these words and phrases <pause> blame myself and change the way say **things** <pause> and the people often construct complex now praises when making their slides and then in discussing them use more natural process <pause> when this thing key phrases jabberwocky also uses the number of samples and tactic transformations <pause> to enumerate the number of ways to my mistake them <pause> so you replied that doesn't sound **like** it really understands be all <pause> troops <pause> i would like in jabberwocky to the tv guide the computer theory conference <pause> his clueless regarding what's been talked about <pause> the still able to follow along <pause> hornet's like alice from lewis carroll said the looking glass after reading the jabberwocky <pause> she said somehow it seems to fill my head with ideas <pause> only at don't know what they are <pause> however someone killed some thing that's clear to anyone

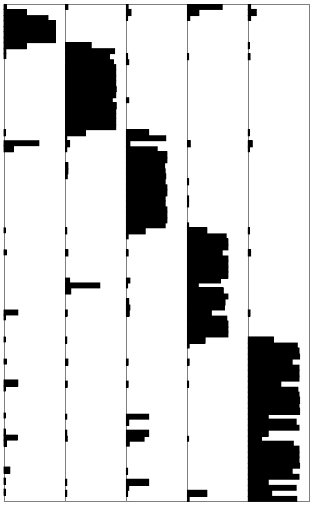


Figure 3: Results from the first experiment.
Across the top, the five slides that the speaker lectured from. On the left, what Jabberwocky heard the speaker say in his narration. On the right, the probability distribution during the course of the talk.

ning IBM ViaVoice and Microsoft PowerPoint.)

8 Using an untrained Jabberwocky

In the first experiment, the speaker lectured from a set of five slides that informally describe the operation of Jabberwocky. Since this is a free-form lecture, Jabberwocky assumes nothing about the order of the slides but uses the probabilistic method to determine what slide to switch to and when. Figure 3 shows the results of the experiment. Across the top are the slides that were lectured from, shown in the order they were lectured from. On the left is what the speaker said (as heard by Jabberwocky.) The words in bold indicate the places where Jabberwocky decided to switch slides. The underlined words were used as evidence for switching to the next slide. On the right is a chart displaying the probability distribution as the talk progressed. Time progresses downward, and the further the solid regions extend to the right, the more confident Jabberwocky was.

Looking at the speaker's narrative (on the left) and comparing it to the slide contents (across the top), we can see that Jabberwocky usually switched to a new slide within five words of when the speaker started discussing it. The five words roughly correspond to two phrases. Looking at the probability distribution (on the right), we see that Jabberwocky was nearly always confident. Isolated spikes indicate places where Jabberwocky was distracted due to the speaker inadvertently saying a key phrase from a different slide. Even though none of these spikes were big enough to cause Jabberwocky to incorrectly change slides, they do suggest that it may be difficult to do large-scale free-form lectures without having unwanted slide changes.

New phrase: "billion stance"
Jabberwocky heard: "... doesn't delicate billion stance be no."
Speaker actually said: "... doesn't sound like it really understands me at all."

New phrase: "clueless forgetting"
Jabberwocky heard: "... theory conference is clueless forgetting what's been talked about."
Speaker actually said: "... theory conference is clueless regarding what's being talked about"

New phrases: "like alice", "lewis carroll", "look glass", "glass factory"
Jabberwocky heard: "... like alice from lewis carroll still looking glass factory..."
Speaker actually said: "... like Alice from Lewis Carroll's "Through the Looking Glass" after reading..."

Figure 4: Results from the second experiment. Interesting phrases learned for the fifth slide.

8.1 Learning through rehearsal

In the second experiment, the speaker rehearsed his presentation with Jabberwocky three times, allowing Jabberwocky to learn the phrases the speaker tended to use in presenting his slides. The speaker used the slides from the first experiment and employed a somewhat consistent narrative in discussing them. The speaker guided Jabberwocky through the talk, using commands like "beginning the next slide" and "beginning the second point" so that Jabberwocky would be able to accurately associate the phrases with the right slide points.

Before the learning began, Jabberwocky had extracted 129 words and phrases from the slides, using the syntactic approaches discussed earlier. From the first rehearsal, Jabberwocky learned 43 new phrases. These consisted largely of phrases taken from speaker transitions, as well as a number of speech recognition errors. From the second rehearsal, Jabberwocky learned 24 new phrases, and rediscovered 14 phrases from the previous rehearsal. From the third rehearsal, Jabberwocky learned 18 new phrases, and rediscovered 23. The increase in rediscovered phrases is due to the speaker's consistency in transition and the speech recognizer's consistency in mis-hearing certain phrases.

Figure 4 shows some interesting phrases that Jabberwocky learned through the course of the rehearsals. The first two ("billion stance" and "clueless forgetting") are due to the speech recognizer mis-hearing the speaker's words. The third set of phrases is due to the speaker's setting up the quotation from "Through the Looking Glass."

9 Related work

Jabberwocky's Bayesian method for skipping around in the slides has much in common with the methods used by Heckerman and Horvitz (Heckerman & Horvitz 1998) for the Microsoft Office Assistant. They take typed free-text queries, extract key terms, and then use a Bayesian approach to determine which of the thousands of help topics will be most useful as an answer to the query. They do almost no grammatical analysis of the text and the probabilities were all human-generated in an impressive knowledge-engineering effort.

Everett, Wauchope and Perez-Quinones (Everett, Wauchope, & Perez-Quinones 1998) build natural language interfaces for virtual reality systems. A challenge they face in one of their VR worlds is that nothing is labeled with names. As a result, different people will often use very different language in referring to the same objects. But, since the number of reasonable commands and queries is very limited, they are able to get good results by simply scanning for key words and then deciding what the person wants based on these key words and on what is around them in the virtual world. Like the Microsoft Office Assistant research, this work is able to use a shallow language understanding because it is able to severely limit the number of things that the user might say.

Allen et al. (Allen *et al.* 1996) deal with noisy speech input to their TRAINS and (the more recent) TRIPS systems. These systems collaborate with a human user to plan routes between cities for various tasks. Unlike the previously described systems, TRAINS-96 actually parses the speech input into speech acts. To eliminate some of speech recognition errors (their speech recognizer was about 70% accurate), the system performs statistical error correction based on a corpus of pre-

viously transcribed dialogues with the system. Then, a parser looks for plausible speech acts in the revised utterance, which are finally filtered and interpreted in the context of what the system and user are currently doing.

Another domain of research in which speech input is used in an interesting way is in intelligent environments. These systems provide multiple modes of inputusually including speech and gesture. But, in contrast to previously described work (which use unconstrained speech input), these systems tend to rely exclusively on command grammars. The Intelligent Room (Coen 1998) is a conference room featuring three display screens, wireless microphones and a dozen cameras (to track the locations of people and to observe gestures). It serves as an intelligent command post for disaster relief planning and as a laboratory tour guide. People can command the room to display information on the display screens and ask questions about what is shown there.

Another interesting intelligent environment, IBM's VizSpace project (Lucente, Zwart, & George 1998), allows a human user to manipulate the objects in its display through a combination of speech and gesture. The user can add, remove, relocate and resize any of a number of graphical objects. This research focuses on the issue of how to combine the speech and gesture information to produce natural interaction.

10 Conclusions

In this paper, we discussed Jabberwocky: a speech-based interface to PowerPoint. We looked at how we hoped to use it in support of many different presentation styles. We also looked at the techniques needed to support these styles: phrase matching, a probabilistic approach and some learning. Jabberwocky is now a usable system (as demonstrated in our first experiment), giving us the opportunity to actually invite people to use it–to see whether it is also useful.

Through these experiences we will learn ways in which to extend Jabberwocky to render it more useful. Some future work we are already aware of concerns looking at non-phrasal clues for going on. For instance, if a speaker finishes the last point of a slide and then pauses, this could be an indication to go on. Also, in analyzing the phrases that Jabberwocky learned through the rehearsal process, it became clear that there are many words that are bad to listen for; monosyllables are nearly always mis-heard and there are many words that are commonly used in paraphrasing. We need to find a principled way of identifying (and rejecting) these words.

One of the triumphs of this research is that we are able to use inaccurate speech recognition input to accurately control a presentation. The reason for this success is that, since Jabberwocky knows what it should hear, it is able to make strong assumptions when it hears things that are not exactly what it expects. This is why we can follow along with presentations that we do not quite understandin observing a lecture at the rocket scientist convention, we can tell when it is the appropriate moment to change the slides. We use our task knowledge to make less knowledge go further.

References

Allen, J.; Miller, B.; Ringger, E.; and Sikorski, T. 1996. Robust understanding in a dialogue system. In *34th Meeting of the Association for Computational Linguistics*.

Bradshaw, S., and Hammond, K. 1999. Constructing indices from citations in collections of research papers. In *62nd Annual Meeting of the American Society for Information Science*.

Carroll, L. 1871. *Through the Looking Glass*.

Coen, M. H. 1998. Design principles for intelligent environments. In *15th National Conference on Artificial Intelligence*.

Everett, S.; Wauchope, K.; and Perez-Quinones, M. 1998. Creating natural language interfaces to vr systems: Experiences, observations and lessons learned. In *4th International Conference on Virtual Systems and Multimedia*.

Franklin, D., and Flachsbart, J. 1998. All gadget and no representation makes jack a dull environment. In *AAAI 1998 Spring Symposium on Intelligent Environments. AAAI TR SS-98-02*.

Franklin, D. 1998. Cooperating with people: The intelligent classroom. In *15th National Conference on Artificial Intelligence*.

Heckerman, D., and Horvitz, E. 1998. Inferring informational goals from free-text queries: A bayesian approach. In *14th Conference on Uncertainty in Artificial Intelligence*.

Lucente, M.; Zwart, G.-J.; and George, A. D. 1998. Visualization space: A testbed for deviceless multimodal user interface. In *AAAI 1998 Spring Symposium on Intelligent Environments. AAAI TR SS-98-02*.

Wilson, D., and Bradshaw, S. 1999. Cbr textuality. In *4th UK Case-Based Reasoning Workshop*.

WordNet. Available and described at http://www.cogsci.princeton.edu/ wn-/w3wn.html.

Xerox. Part of speech tagger. available at parcftp-.xerox.com/pub/tagger/tagger-1-0.tar.z and described at http://kb.rxrc.xerox.com-/grenoble/mltt/fsNLP/tagger.html.