IMPROVING EXTRACTIVE DIALOGUE SUMMARIZATION BY UTILIZING HUMAN FEEDBACK

Margot Mieskes, Christoph Müller and Michael Strube
EML Research gGmbH
Heidelberg
Germany
`http://www.eml-research.de/nlp`

ABSTRACT
Automatic summarization systems usually are trained and evaluated in a particular domain with fixed data sets. When such a system is to be applied to slightly different input, labor- and cost-intensive annotations have to be created to retrain the system. We deal with this problem by providing users with a GUI which allows them to correct automatically produced imperfect summaries. The corrected summary in turn is added to the pool of training data. The performance of the system is expected to improve as it adapts to the new domain.

KEY WORDS
Multi-Party Dialogues, Automatic Summarization, GUI, Feedback, Learning

1 Introduction

To write meeting minutes is a task which almost nobody likes to do. The antipathy towards summarizing meetings manually has several reasons. Basically there are two standard situations: First, a meeting participant is assigned the task. In this case the person in charge of summarizing is not able to properly participate in the meeting. Second, an outsider is asked to summarize the meeting. However, such a person often does not have sufficient knowledge needed to identify the important parts. This increases the workload and typically interrupts the discussion, as the person taking notes has to ask whether something was important or not. The leader of the meeting and the recorder could also meet in advance and set the agenda, agree on signals etc. In both scenarios there is a lot of work necessary after the meeting to write a summary based on the notes taken during the meeting itself.

The aim of our automatic meeting summarization system is to offer a tool to support creating extractive summaries. But instead of providing the user with a ready-made system, where everything is fixed and settled, we base our summarization on the concept of an interactive agent. This is based on the idea that in order ”to be truly helpful, an assistant must learn over time, through interacting with the user and its environment – otherwise it will only repeat its mistakes” (Lieberman & Maulsby, 1996). Interactive agents have been applied to various tasks in the past (Middleton, 2001), but to our knowledge not to the task of summarizing meetings.

The disadvantage of ready-made summarization systems – and many natural language processing systems in general – is that they rely on a predefined set of training data which cannot be easily exchanged or extended by adding data suitable for the domain of interest. We overcome this problem by providing the user with an interface which allows them to correct the automatically produced summary. The correction is fed back to the system, thus making it possible for the system to adapt to new domains over time. This strategy allows to start out with a smaller amount of training data. The quality of the summarization gradually improves as more high quality training data of the right domain become available. For development the system is based on the ICSI Corpus, which contains 75 manually transcribed dialogues, each on average one hour long, and with between three and 13 participants (Janin et al., 2003).

We will give an outline of the current state of the art in summarization systems in Section 2. we will present a system that is easy to use even for people who are not very adept in using complicated systems and we will show how much control the system and the user will have (Middleton, 2001). Furthermore, we will describe what the initial state of the system is, how this initial state is reached and how the feedback is provided in order to improve the output results (Sections 3, 4 and 5).

2 Available Summarization Systems

We will briefly discuss summarization systems which are either available for download or accessible as demos through the web. In general, these systems are not intended to be used by naive users. Either the configuration options are hidden in some arcane configuration file or the interface is not intuitive, because it was intended for the developers.

One of the earliest systems is *SUMMARIST* (Hovy & Lin, 1999), which produces generative instead of only extractive summaries of news articles. It combines machine learning methods (e.g. decision trees), heuristics (e.g. title and query word matching) and external knowledge sources such as WordNet (Fellbaum, 1998) for extracting relevant parts of the document. The extraction is used as an intermediate step, from which the summary is generated. After the topic discussed in the text is determined, the text is analysed and reinterpreted in order to compact and fuse the extracted elements (e.g ”he bought pears, apples and bananas” is fused to ”he bought fruit”). The generation itself is based on two components: the microplanner and the sentence generator. The microplanner builds the structure of the text based on the information from the topic identification. It already selects important words, main verbs, principal theme and focus. The sentence generator fills the structure with grammatical sentences. The system was evaluated in the context of the SUMMAC evaluation, where several shortcomings became obvious. First, the amount of training data was not sufficient. Second, there was a lack of world knowledge, which is necessary for topic identification. And finally, a shortage in resources to further enhance the microplanner became apparent. A big difference between *SUMMARIST* and our system is that it summarizes text instead of multiparty dialogue. The tool is available as a demo on the web. The interface is very complicated and not very intuitive to use.

The system MEAD (Radev et al., 2004) was also developed for text summarization, but covers both single and multi document summarization and can handle several languages. It also offers several varieties of extraction methods and contains evaluation metrics like *kappa* (Carletta, 1996) and *precision/recall* (Baeza-Yates & Ribeiro-Neto, 1999). The process of summarizing consists of four steps: first the texts are converted into the MEAD format. Second, from each sentence a set of features is extracted. Third, these features result in a score for each sentence. These scores are then refined using cross-sentence dependencies. The system allows the developer to access many options. There are many ways to customize the summaries. These options and customizations have to be set in a configuration file. The system is mainly built for usage on the command line and apparently not intended for naive users.

Finally, DiaSumm (Zechner & Waibel, 2000) is closest to our system, because it can be applied not only to texts but can also summarize dialogues and meetings. The meeting summarization is embedded in the *Meeting Browser*[1]. During preprocessing a large variety of annotations is added, but most of them depend on manual annotation. Few have been implemented to work automatically as well, although methods to develop automatic annotations are outlined. The extraction is based on *Maximum Marginal Relevance* (Carbonell & Goldstein, 1998), the detection of question-answer pairs, and other features. Intrinsic evaluation is performed based on word counts and summary accuracy. The system was built to summarize telephone dialogues. Whether it is applicable to meetings with more than two participants is unclear.

The Meeting Browser is a powerful tool, which not only allows the transcription of the meetings to be viewed, but also video recordings. DiaSumm provides summaries for a quick overview over the meeting. Unfortunately, there is no indication about where and how to obtain the tool. The description also mentions search functions to look for keywords and topics. One of the features is described as ”Create and customize dialogue, audio, and video summaries to the user’s particular needs”, but details are not given[2].

None of the systems allows the user to intervene in the summarization. All mentioned systems are just black boxes where the summary comes out and the user has to be happy with the result. Additionally, none of the systems has been tested on multi-party dialogues or meetings. Most have been developed for text or news speech (Hori et al., 2002) or dialogues as presented above. But summarizing multi-party dialogues gives rise to additional problems, which cannot be dealt only with methods from text summarization (McKeown et al., 2005).

3 Annotation

As already mentioned, our system creates extractive summaries by selecting and concatenating relevant elements of a meeting transcript. One question that we had to address was the size of these elements, and thus the granularity of the summaries. The original ICSI Meeting corpus is structured as a sequence of semi-automatically created segments. These segments are not intended to capture any linguistically relevant elements like *turn* or *sentence*, which is mainly due to the fact that the definition of e.g. *turn* is difficult in the context of multi-party dialogue with a considerable amount of overlapping speech.

As a rule of thumb, the developers of the ICSI corpus inserted a break in the current speaker’s segment whenever some other speaker started to talk. This principle, however, was not followed consequently, as it would have led to a high degree of fragmentation (Janin, 2002). As a result, the corpus contains both long segments for speaker contributions that clearly do overlap with others, but at the same time there are also longer, uninterrupted contributions by one speaker which are split into several segments.

Despite these irregularities, we decided to use the original segments, since they are the major structuring elements in the corpus. Alternatively, we also experimented with so-called *spurts*, i.e. sequences of segments with pauses no longer than 500 milliseconds. However, a pilot annotation experiment proved spurts to be inappropriate for extractive summarization since they tend to become extremely long and thus too coarse-grained.

In the manual annotation phase, our annotators used the annotation tool MMAX2 (Müller & Strube, 2006) to mark segments as relevant or not relevant for a summary. In MMAX2, annotation data is stored in the form of so-called *markables*. Markables pertaining to the same phenomenon (e.g. segments, but also things like Part-of-Speech (POS)

[1] `http://penance.is.cs.cmu.edu/meeting_room/index.html`

[2] `http://penance.is.cs.cmu.edu/meeting_room/index.html`

tags or syntactic chunks) are grouped on a dedicated *annotation level*. Arbitrarily many annotation levels can coexist for the same underlying data, because each markable references the words that it applies to by means of a pointer into a separate file containing the words.

One fundamental principle of multi-level annotation is that data from diverse, unrelated levels can (and in fact should) be kept apart. Therefore, while the granularity of the annotation of elements to be included in the summary is defined by segments, this information should not be stored on the segment level itself, but on a dedicated *summary* level which contains markables which can be mapped to segments one-to-one.

4 System Architecture

Figures 1 and 2 show an overview of all elements of our summarization system and the steps involved in the project. Figure 1 shows the procedure during the development of the system. Figure 2 shows the system during usage.

In both figures square components represent data that can be further processed or used. Round components represent actions performed on the data that is fed to the component. For example, the annotated transcriptions (square) are split into training and test data (both square). The machine learning (round) component produces a model (square) from the training and test data. This model is then used for producing summaries. The lighter coloured elements have already been implemented or, like the summarizaton component, are currently being implemented. All steps presented here will be addressed in more detail in the following.

Figures 1 and 2 have some common components, namely the speech recognizer, the preprocessing, the pronoun and anaphora resolution and the annotation component. During the development (Figure 1) the data from these various stages is gathered manually. The data serves two goals: First, it is used to *simulate* the output of automatic components during development. Second, it is used to *build* these components. POS data e.g. is used to retrain existing taggers. These common components are the following:

First, a speech recognizer would be most important. So far, we have relied on manual transcriptions of the ICSI Meeting Corpus, which were provided with the sound data. We aim to add a speech recognizer in the near future. The output of the speech recognizer will be transcribed speech.

Second, various preprocessing steps are applied. Among these steps are POS tagging, disfluency detection and dialogue act annotation, all of which have already been implemented. When we add the speech recognizer in the near future, we will also need a sentence boundary detection component.

A third step in summarizing meetings is pronoun and anaphora resolution. Spoken language contains a larger amount of pronouns than written texts (Strube & Müller, 2003). In order to improve extraction recall and to avoid confusing and unclear summaries, these pronouns have to be resolved. This is currently being developed.

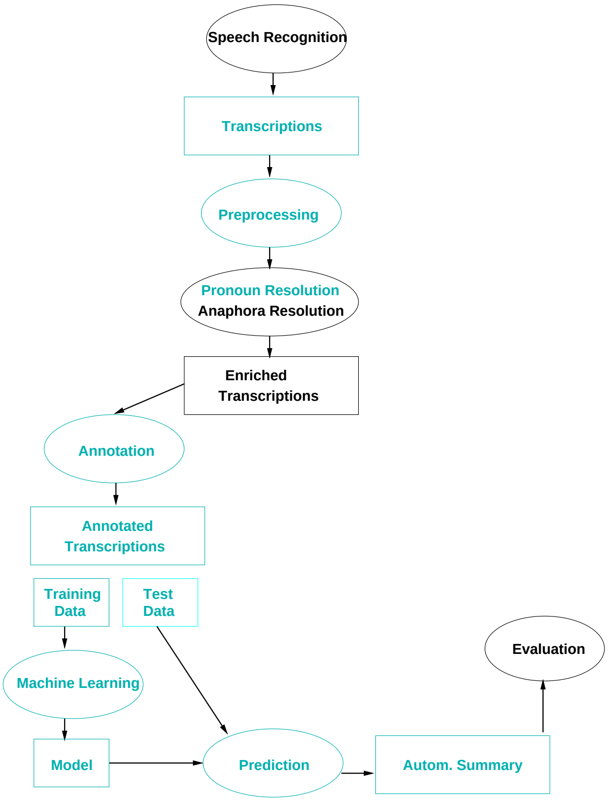


Figure 1. Overview of the Development System

A fourth step is topic boundary detection. The meetings in our corpus are quite long (in average one hour) and in most cases several topics are discussed, which have to be summarized separately. As there are no agendas for the meetings, the topic boundaries have to be found automatically. This task is done by a reliable method already developed.

4.1 Development System

The final step in this chain is the detection of summary relevant items. These items are manually annotated. The manual data is split into training and test data. The first serves as input for a machine learning method (e.g. Decision Trees), which creates a model. This model then is tested on the test data and if necessary the parameters for the training step are adjusted. Currently, we are using a total of 12 meetings to build the automatic annotation steps. 9 meetings are used for training and 3 for testing. The final system is then evaluated for its quality, e.g. with precision/recall or other methods described in (Zechner, 2002). This work is currently carried out.

It is important to note that there is a considerable difference between the ”enriched transcriptions” and the ”annotated transcriptions” in Figures 1 and 2. The enriched

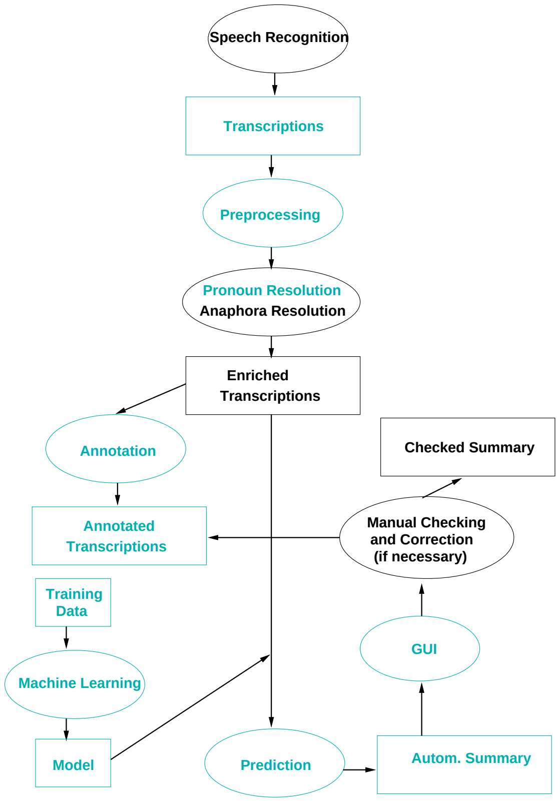


Figure 2. Overview of the System when Used

transcriptions are necessary for the further processing, but they are by no means sufficient for summarization. The annotated transcriptions provide topic boundary information as well as information about summary relevant items.

4.2 The Interactive Production System

Figure 2 shows the system as it is supposed to work during usage. All stages which were done manually in the development stage are now performed automatically. The model is directly applied to the transcribed data and the annotation for summary relevant items is based on the output from the model. This automatic summary can now be checked via a graphical user interface (GUI). The user can add or remove items from the summary. The information about new or changed items is then fed back to the available annotations and used to retrain the model and improve future automatic summaries. Depending on the environment in which the system is used, some additional training is needed, but it will learn from the interaction with the user and improve the results. The details about the GUI will be explained in Section 5 below.

During the development of the summarization system we are currently experimenting with various summarization methods (e.g. tf*idf, MMR (Carbonell & Goldstein, 1998), Lexical Chains (Barzilay & Elhadad, 1999)), but also methods based on machine learning. The method that proves most successful and best applicable to the interactive system will be incorporated into the system as shown in Figure 2.

The summarization system as a whole will be used as a plugin in the MMAX2 tool, as described in Section 5. The system will provide a summary based on the knowledge gathered so far. This summary is then presented to the user, who can approve it or change it. If the summary is approved it can be saved for further usage. If the user want to change it, they can do so by changing the attribute of the elements in the summary to *non-relevant*. If he wants to add elements from the meeting to the summary, he can do so by changing the attribute of elements in the meeting from *non-relevant* to *relevant*, as shown in Figure 3. These elements are then added to the summary. Once the summary is changed it can be saved for further usage. The changed summaries are also used to update the model for producing the initial summaries.

5 The Graphical User Interface and its Combination with MMAX2

The data format described in Section 3 is not only used within the MMAX2 annotation tool. There is also a MMAX2 API which supplies Java bindings for data elements like markables, annotation levels, or entire discourses, among others. Our summarization tool takes the form of a plugin to MMAX2. As such, it can use the MMAX2 API to produce output which is structurally identical to the data produced by the manual annotation, and which can be viewed within the same annotation tool.

Support for Java plugins is a feature that has recently been added to MMAX2. A plugin is a user-specified Java class that can be accessed from within the tool via the Plugins menu. In order for a Java class to be usable as a plugin, three things are required.

1. The Java class has to be derived from the class org.eml.MMAX2.MMAX2Plugin. In addition, the class can implement arbitrary methods to be executed on demand. The first parameter for each of these methods must be an object of the class MMAX2Discourse, followed by an arbitrary number of string attributes.

2. For each plugin method that should be callable from the MMAX2 Plugins menu, an entry must exist in the global plugins.xml file. This file has the following structure:

```
<?xml version="1.0"?>
<plugins>
 <plugin name="Summarization GUI"
         class="summGUI.SummGUI">
  <parameter attribute="task"
             value="showSummAgent"/>
 </plugin>
</plugins>
```

3. The Java class must be in the classpath of MMAX2.

The plugins.xml file must be located in the MMAX2 installation directory. At startup, an entry in the Plugins menu is created for every plugin method definition found in this file.

Once a plugin method has been called, it has access to the same data as the MMAX2 instance from which it was called. This means that changes to the data in one component are visible to the other, and vice versa. In addition, the plugin can call methods on individual markable objects to keep the annotation tool display synchronized, i.e. to highlight a markable in the tool's display. If necessary, the display is first scrolled to the position of the markable.

Figure 3 shows the main window of MMAX2 on the left side and the summary plugin window on the left side. In the main window the segments are shown together with the speakers (e.g. fe016). Some segments are very short, some are quite long. Words in light grey are non-speech elements, like noise or transcribers' comments. The elements that have been annotated as relevant for the summary are highlighted with a grey background and bold font.

The summarization plugin window on the right side of Figure 3 shows all segments that were marked as relevant on the summary level. Topic breaks (double lines) are marked here as well, to give the reader more information on the division of the meeting. As the plugin is synchronized with the MMAX2 display, elements in the summary can be selected and the main window shows the same element in its context. This is also shown in Figure 3. In the main window the attribute of the selected segment can now be changed from *relevant* to *non-relevant* and vice versa, thus removing elements from or adding elements to the summary.

As shown in Figure 2 above the corrected summary can then be fed back to the machine learning component which updates the model according to the new training data.

6 Conclusions and Future Work

We presented ongoing work on a system for automatic extractive meeting summarization. Unlike other systems, our system is not just a black box, where the user is unable to influence the production of the summary. Rather, our system allows the user to actively improve the summaries via a GUI-based feedback loop. Compared to other systems for automatic summarization, it is designed for an end-user rather than a developer. As such, the control mechanisms are simple and straightforward.

In future work we would like to evaluate the usability of the system with naive users who deal with meetings and their summaries as part of their daily work. Before this extrinsic evaluation can take place, an intrinsic evaluation on the development system has to be done. For this, a suitable and updateable method for automatically extracting relevant segments from the meetings has to be found and incorporated into the system.

Also, an addition to both the development and usage system is planned: A method to save a summary once it is corrected. For the developer this should allow for comparing various steps during the development process and observing the development of the initial model. For the end-user, the system should allow for two functionalities: First, the user can load an already saved summary. Second, the system could keep track of saved summaries, and once it recognizes an already summarized meeting, it can offer to load an existing summary or create a new one.

By presenting this system we showed that not all data-driven applications in NLP rely on a predefined set of training data, and that there are applications which are not just black boxes outside the control of the user. Our system is designed to be able to extend an initially limited amount of training data to become more and more suitable for the domain of interest and the context in which it is used. This is achieved by providing a GUI-based tool for collecting and utilizing user feedback. The feasibility of such a bootstrapping approach is mainly due to the nature of the application: For (meeting) summarization, there is a practical application scenario in which a linguistically naive domain expert evaluates and corrects automatically created summaries as a part of his or her job, thus producing improved training data as a by-product. For other NLP applications like e.g. parsing, the situation is different: Parsing natural language is not a practical application in its own right, and even if parser output is manually evaluated and corrected (e.g. in the course of a research project), this has to be done by a linguistics expert, which puts a narrow limit on the amount of data that can be produced.

However, in the future similar applications in natural language processing may emerge which can be dealt with in this way.

Acknowledgments

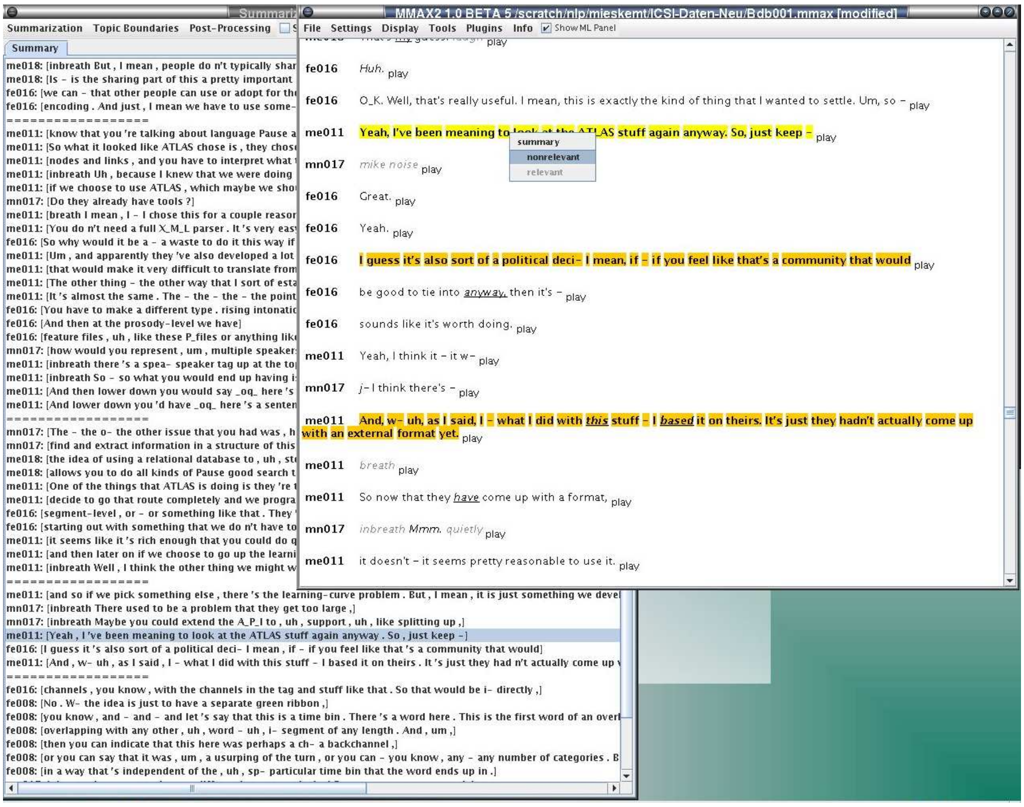
This work has been supported by the DFG under grant STR 545/2-1 within the DIANA-Summ Project and by the Klaus Tschira Foundation.

References

Baeza-Yates, R. & B. Ribeiro-Neto (1999). *Modern Information Retrieval*. New York, NY, Harlow, UK: ACM Press, Pearson Addison-Wesley.

Barzilay, R. & M. Elhadad (1999). Using lexical chains for text summarization. In I. Mani & M. T. Maybury (Eds.), *Advances in Automatic Text Summarization*, pp. 111–121. Cambridge/MA, London/England: MIT Press.

Carbonell, J. G. & J. Goldstein (1998). The use of MMR, diversity-based reranking for reordering documents and producing summaries. In A. Moffat & J. Zobel (Eds.), *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval,* Melbourne, Australia, pp. 335–336.



```
me018: [inbreath But , I mean , people do n't typically shar
me018: [Is - is the sharing part of this a pretty important
fe016: [we can - that other people can use or adopt for th
fe016: [encoding . And just , I mean we have to use some-
====================
me011: [know that you 're talking about language Pause a
me011: [So what it looked like ATLAS chose is , they chose
me011: [nodes and links , and you have to interpret what
me011: [inbreath Uh , because I knew that we were doing
me011: [if we choose to use ATLAS , which maybe we sho
mn017: [Do they already have tools ?]
me011: [breath I mean , I - I chose this for a couple reasor
me011: [You do n't need a full X_M_L parser . It 's very eas
fe016: [So why would it be a - a waste to do it this way if
me011: [Um , and apparently they 've also developed a lot
me011: [that would make it very difficult to translate from
me011: [The other thing - the other way that I sort of esta
me011: [It 's almost the same . The - the - the - the point
fe016: [You have to make a different type . rising intonatic
fe016: [And then at the prosody-level we have]
fe016: [feature files , uh , like these P_files or anything like
mn017: [how would you represent , um , multiple speakers
me011: [inbreath there 's a spea- speaker tag up at the to
me011: [inbreath So - so what you would end up having i
me011: [And then lower down you would say _oq_ here 's
me011: [And lower down you 'd have _oq_ here 's a senten
====================
mn017: [The - the o- the other issue that you had was , h
mn017: [find and extract information in a structure of this
me018: [the idea of using a relational database to , uh , st
me018: [allows you to do all kinds of Pause good search t
me011: [One of the things that ATLAS is doing is they 're t
me011: [decide to go that route completely and we progra
fe016: [segment-level , or - or something like that . They
fe016: [starting out with something that we do n't have to
me011: [it seems like it 's rich enough that you could do q
me011: [and then later on if we choose to go up the learni
me011: [inbreath Well , I think the other thing we might w
====================
me011: [and so if we pick something else , there 's the learning-curve problem . But , I mean , it is just something we devel
mn017: [inbreath There used to be a problem that they get too large ,]
mn017: [inbreath Maybe you could extend the A_P_I to , uh , support , uh , like splitting up ,]
me011: [Yeah , I 've been meaning to look at the ATLAS stuff again anyway . So , just keep -]
fe016: [I guess it 's also sort of a political deci- I mean , if - if you feel like that 's a community that would]
me011: [And , w- uh , as I said , I - what I did with this stuff - I based it on theirs . It 's just they had n't actually come up
====================
fe016: [channels , you know , with the channels in the tag and stuff like that . So that would be i- directly ,]
fe008: [No . W- the idea is just to have a separate green ribbon ,]
fe008: [you know , and - and - and let 's say that this is a time bin . There 's a word here . This is the first word of an over
fe008: [overlapping with any other , uh , word - uh , i- segment of any length . And , um ,]
fe008: [then you can indicate that this here was perhaps a ch- a backchannel ,]
fe008: [or you can say that it was , um , a usurping of the turn , or you can - you know , any - any number of categories . B
fe008: [in a way that 's independent of the , uh , sp- particular time bin that the word ends up in .]
```

fe016 *Huh.* play

fe016 O_K. Well, that's really useful. I mean, this is exactly the kind of thing that I wanted to settle. Um, so – play

me011 **Yeah, I've been meaning to look at the ATLAS stuff again anyway. So, just keep –** play

mn017 *mike noise* play

fe016 Great. play

fe016 Yeah. play

fe016 **I guess it's also sort of a political deci– I mean, if – if you feel like that's a community that would** play

fe016 be good to tie into *anyway,* then it's – play

fe016 sounds like it's worth doing. play

me011 Yeah, I think it – it w– play

mn017 *j*– I think there's – play

me011 **And, w– uh, as I said, I – what I did with *this* stuff – I *based* it on theirs. It's just they hadn't actually come up with an external format yet.** play

me011 *breath* play

me011 So now that they *have* come up with a format, play

mn017 *inbreath Mmm. quietly* play

me011 it doesn't – it seems pretty reasonable to use it. play

Figure 3. MMAX2 Main Window and the Summary Plugin

Carletta, J. (1996). Assessing agreement on classification tasks: The kappa statistic. *Computational Linguistics*, 22(2):249–254.

Fellbaum, C. (Ed.) (1998). *WordNet: An Electronic Lexical Database*. Cambridge, Mass.: MIT Press.

Hori, C., S. Furui, R. Malkin, H. Yu & A. Waibel (2002). Automatic summarization of English broadcast news speech. In *Proceedings of the Human Language Technology Conference*, San Diego, Cal., 24–27 March 2002, pp. 241–246.

Hovy, E. & C.-Y. Lin (1999). Automated text summarization in SUMMARIST. In I. Mani & M. T. Maybury (Eds.), *Advances in Automatic Text Summarization*, pp. 82–94. Cambridge/MA, London/England: MIT Press.

Janin, A. (2002). Meeting recorder. In *Proceedings of the Applied Voice Input/Output Society Conference (AVIOS)*, San José, California, USA, May 2002.

Janin, A., D. Baron, J. Edwards, D. Ellis, D. Gelbart, N. Morgan, B. Peskin, T. Pfau, E. Shriberg, A. Stolcke & C. Wooters (2003). The ICSI meeting corpus. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, Hong Kong, 6-10 April 2003, pp. 364–367.

Lieberman, H. & D. Maulsby (1996). Instructable agents: Software that just keeps getting better. *IBM Systems Journal*, 35(3–4).

McKeown, K. R., J. Hirschberg, M. Galley & S. Maskey (2005). From text to speech summarization. In *IEEE International Conference on Acoustics, Speech and Signal Processing; Special Session on Human Language Technology: Applications and Challenges for Speech Processing*, Philadelphia, USA, March 18–23 2005.

Middleton, S. E. (2001). *Interactive agents: A review of the field*. Technical Report ECSTR-IAM01-001, ISBN: 0854327320,: University of Southampton.

Müller, C. & M. Strube (2006). Multi-level annotation of linguistic data with MMAX2. In S. Braun, K. Kohn & J. Mukherjee (Eds.), *Corpus Technology and Language Pedagogy: New Resources, New Tools, New Methods*. Frankfurt a.M., Germany: Peter Lang.

Radev, D., T. Allison, S. Blair-Goldensohn, J. Blitzer, A. Celibi, S. Dimitrov, E. Drabek, A. Hakim, W. Lam, D. Liu, J. Otterbacher, H. Qi, H. Saggion, S. Teufel, M. Topper, A. Winkel & Z. Zhang (2004). MEAD – a platform for multidocument multilingual text summarization. In *Proceedings of the 4th International Conference on Language Resources and Evaluation*, Lisbon, Portugal, 26–28 May, 2004.

Strube, M. & C. Müller (2003). A machine learning approach to pronoun resolution in spoken dialogue. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, Sapporo, Japan, 7–12 July 2003, pp. 168–175.

Zechner, K. (2002). Automatic summarization of open-domain multiparty dialogues in diverse genres. *Computational Linguistics*, 28(4):447–485.

Zechner, K. & A. Waibel (2000). Diasumm: Flexible summarization of spontaneous dialogues in unrestricted domains. In *Proceedings of the 18th International Conference on Computational Linguistics*, Saarbrücken, Germany, 31 July – 4 August 2000, pp. 968–974.