ACL-08: HLT

46th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies

Proceedings of the Demo Session

June 16, 2008
The Ohio State University
Columbus, Ohio, USA

Production and Manufacturing by
Omnipress Inc.
2600 Anderson Street
Madison, WI 53707
USA

Order copies of this and other ACL proceedings from:

Association for Computational Linguistics (ACL)
209 N. Eighth Street
Stroudsburg, PA 18360
USA
Tel: +1-570-476-8006
Fax: +1-570-476-0860
acl@aclweb.org

Introduction

Welcome to the proceedings of the demo session. We received 21 submissions, 9 of which were selected for inclusion in the program after review by at least two members of the program committee.

Chair:

Jimmy Lin (University of Maryland)

Program Committee:

Marine Carpuat (HKUST)
Joyce Chai (Michigan State University)
Trevor Cohn (University of Edinburgh)
Damianos Karakos (Johns Hopkins University)
Terry Koo (MIT)
Alberto Lavelli (FBK-irst)
Don Metzler (Yahoo! Research)
Andrew Rosenberg (Columbia University)
Hiroshi Nakagawa (University of Tokyo)
Takenobu Tokunaga (Tokyo Instiute of Technology)

Table of Contents

Demonstration of a POMDP Voice Dialer

Jason Williams
AT&T Labs – Research, Shannon Laboratory
180 Park Ave., Florham Park, NJ 07932, USA
jdw@research.att.com

Abstract

This is a demonstration of a voice dialer, implemented as a partially observable Markov decision process (POMDP). A real-time graphical display shows the POMDP's probability distribution over different possible dialog states, and shows how system output is generated and selected. The system demonstrated here includes several recent advances, including an action selection mechanism which unifies a hand-crafted controller and reinforcement learning. The voice dialer itself is in use today in AT&T Labs and receives daily calls.

1 Introduction

Partially observable Markov decision processes (POMDPs) provide a principled formalism for planning under uncertainty, and past work has argued that POMDPs are an attractive framework for building spoken dialog systems (Williams and Young, 2007a). POMDPs differ from conventional dialog systems in two respects. First, rather than maintaining a single hypotheses for the dialog state, POMDPs maintain a probability distribution called a *belief state* over many possible dialog states. A distribution over a multiple dialog state hypotheses adds inherent robustness, because even if an error is introduced into one dialog hypothesis, it can later be discarded in favor of other, uncontaminated dialog hypotheses. Second, POMDPs choose actions using an optimization process, in which a developer specifies high-level goals and the optimization works out the detailed dialog plan. Because of these innovations, POMDP-based dialog systems have, in research settings, shown more resilience to speech recognition errors, yielding shorter dialogs with higher task completion rates (Williams and Young, 2007a; Williams and Young, 2007b).

Because POMDPs differ significantly from conventional techniques, their operation can be difficult to conceptualize. This demonstration provides an accessible illustration of the operation of a state-of-the-art POMDP-based dialog system. The system itself is a voice dialer, which has been operational for several months in AT&T Labs. The system incorporates several recent advances, including efficient large-scale belief monitoring (akin to Young et al., 2006), policy compression (Williams and Young, 2007b), and a hybrid hand-crafted/optimized dialog manager (Williams, 2008). All of these elements are depicted in a graphical display, which is updated in real time, as a call is progressing. Whereas previous demonstrations of POMDP-based dialog systems have focused on showing the probability distribution over dialog states (Young et al., 2007), this demonstration adds new detail to convey how actions are chosen by the dialog manager.

In the remainder of this paper, Section 2 presents the dialog system and explains how the POMDP approach has been applied. Then, section 3 explains the graphical display which illustrates the operation of the POMDP.

2 System description

This application demonstrated here is a voice dialer application, which is accessible within the AT&T research lab and receives daily calls. The dialer's vo-

Proceedings of the ACL-08: HLT Demo Session (Companion Volume), pages 1–4,
Columbus, June 2008.

cabulary consists of 50,000 AT&T employees.

The dialog manager in the dialer is implemented as a POMDP. In the POMDP approach, a distribution called a belief state is maintained over many possible dialog states, and actions are chosen using reinforcement learning (Williams and Young, 2007a). In this application, a distribution is maintained over all of the employees' phone listings in the dialer's vocabulary, such as Jason Williams' office phone or Srinivas Bangalore's cell phone. As speech recognition results are received, this distribution is updated using probability models of how users are likely to respond to questions and how the speech recognition process is likely to corrupt user speech. The benefit of tracking this belief state is that it synthesizes all of the ASR N-Best lists over the whole dialog – i.e., it makes the most possible use of the information from the speech recognizer.

POMDPs then choose actions based on this belief state using reinforcement learning (Sutton and Barto, 1998). A developer writes a reward function which assigns a real number to each state/action pair, and an optimization algorithm determines how to choose actions in order to maximize the expected *sum* of rewards. In other words, the optimization performs planning and this allows a developer to specify the trade-off to use between task completion and dialog length. In this system, a simple reward function assigns -1 per system action plus +/- 20 for correctly/incorrectly transferring the caller at the end of the call. Optimization was performed roughly following (Williams and Young, 2007b), by running dialogs in simulation.

Despite their theoretical elegance, applying a POMDP to this spoken dialog system has presented several interesting research challenges. First, scaling the number of listings quickly prevents the belief state from being updated in real-time, and here we track a distribution over *partitions*, which is akin to a beam search in ASR (Young et al., 2006). At first, all listings are undifferentiated in a single master partition. If a listing appears on the N-Best list, it is separated into its own partition and tracked separately. If the number of partitions grows too large, then low-probability partitions are folded back into the master undifferentiated partition. This technique allows a well-formed distribution to be maintained over an arbitrary number of concepts in real-time.

Second, the optimization process which chooses actions is also difficult to scale. To tackle this, the so-called "summary POMDP" has been adopted, which performs optimization in a compressed space (Williams and Young, 2007b). Actions are mapped into clusters called *mnemonics*, and states are compressed into state feature vectors. During optimization, a set of template state feature vectors are sampled, and values are computed for each action mnemonic at each template state feature vector.

Finally, in the classical POMDP approach there is no straightforward way to impose rules on system behavior because the optimization algorithm considers taking any action at any point. This makes it impossible to impose design constraints or business rules, and also needlessly re-discovers obvious domain properties during optimization. In this system, a hybrid POMDP/hand-crafted dialog manager is used (Williams, 2008). The POMDP and conventional dialog manager run in parallel; the conventional dialog manager nominates a *set* of one *or more* allowed actions, and the POMDP chooses the optimal action from this set. This approach enables rules to be imposed and allows prompts to easily be made context-specific.

The POMDP dialer has been compared to a convention version in dialog simulation, and improved task completion from 92% to 97% while keeping dialog length relatively stable. The system has been deployed in the lab and we are currently collecting data to assess performance with real callers.

3 Demonstration

A browser-based graphical display has been created which shows the operation of the POMDP dialer in real time, shown in Figure 1. The page is updated after the user speech has been processed, and before the next system action has been played to the user. The left-most column shows the system prompt which was just played to the user, and the N-Best list of recognized text strings, each with its confidence score.

The center column shows the POMDP belief state. Initially, all of the belief is held by the master, undifferentiated partition, which is shown as a green bar and always shown first. As names are recognized, they are tracked separately, and the top 10

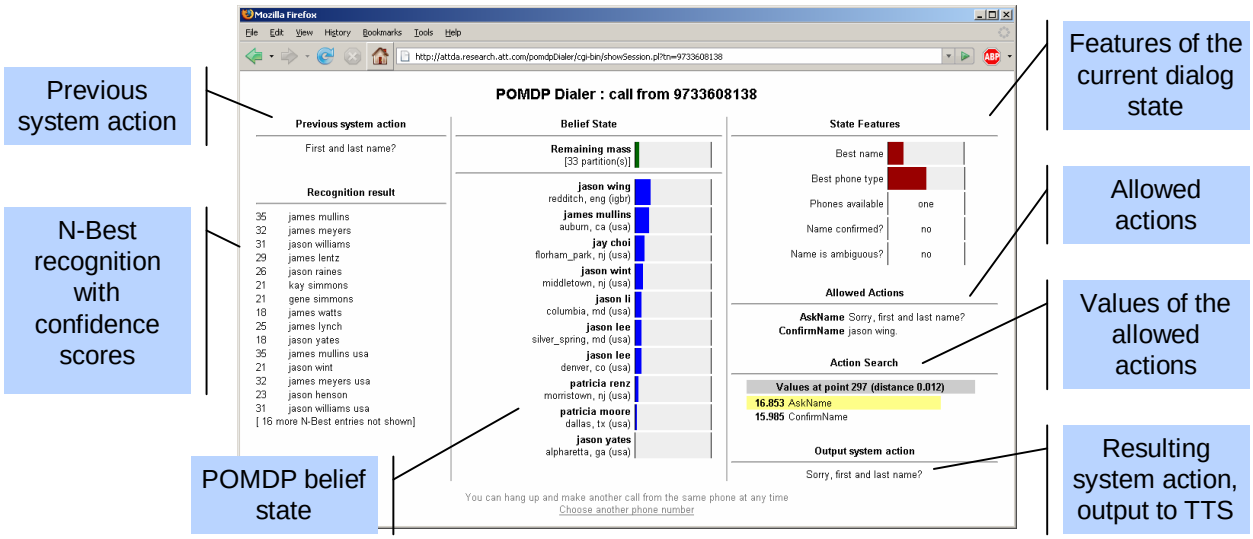


Figure 1: *Overview of the graphical display. Contents are described in the text.*

names are shown as blue bars, sorted by their belief. If the system asks for the phone type (office or mobile), then the bars sub-divide into a light blue (for office) and dark blue (for mobile).

The right column shows how actions are selected. The top area shows the features of the current state used to choose actions. Red bars show the two continuous features: the belief in the most likely name and most likely type of phone. Below that, three discrete features are shown: how many phones are available (none, one, or both); whether the most likely name has been confirmed (yes or no); and whether the most likely name is ambiguous (yes or no). Below this, the allowed actions (i.e., those which are nominated by the hand-crafted dialog manager) are shown. Each action is preceded by the action mnemonic, shown in bold. Below the allowed actions, the action selection process is shown. The values of the action mnemonic at the closest template point are shown next to each action mnemonic. Finally the text of this action, which is output to the caller, is shown at the bottom of the right-hand column. Figure 2 shows the audio and video transcription of an interaction with the demonstration.

4 Conclusion

This demonstration has shown the operation of a POMDP-based dialog system, which incorporates recent advances including efficient large-scale belief monitoring, policy compression, and a unified hand-crafted/optimized dialog manager. A graphical display shows the operation of the system in real-time, as a call progresses, which helps make the POMDP approach accessible to a non-specialist.

Acknowledgments

Thanks to Iker Arizmendi and Vincent Goffin for help with the implementation.

References

R Sutton and A Barto. 1998. *Reinforcement Learning: an Introduction*. MIT Press.

JD Williams and SJ Young. 2007a. Partially observable Markov decision processes for spoken dialog systems. *Computer Speech and Language*, 21(2):393–422.

JD Williams and SJ Young. 2007b. Scaling POMDPs for spoken dialog management. *IEEE Trans. on Audio, Speech, and Language Processing*, 15(7):2116–2129.

JD Williams. 2008. The best of both worlds: Unifying conventional dialog systems and POMDPs. In *(In submission)*.

SJ Young, JD Williams, J Schatzmann, MN Stuttle, and K Weilhammer. 2006. The hidden information state approach to dialogue management. Technical Report CUED/F-INFENG/TR.544, Cambridge University Engineering Department.

SJ Young, J Schatzmann, B R M Thomson, KWeilhammer, and H Ye. 2007. The hidden information state dialogue manager: A real-world POMDP-based system. In *Proc NAACL-HLT, Rochester, New York, USA*.

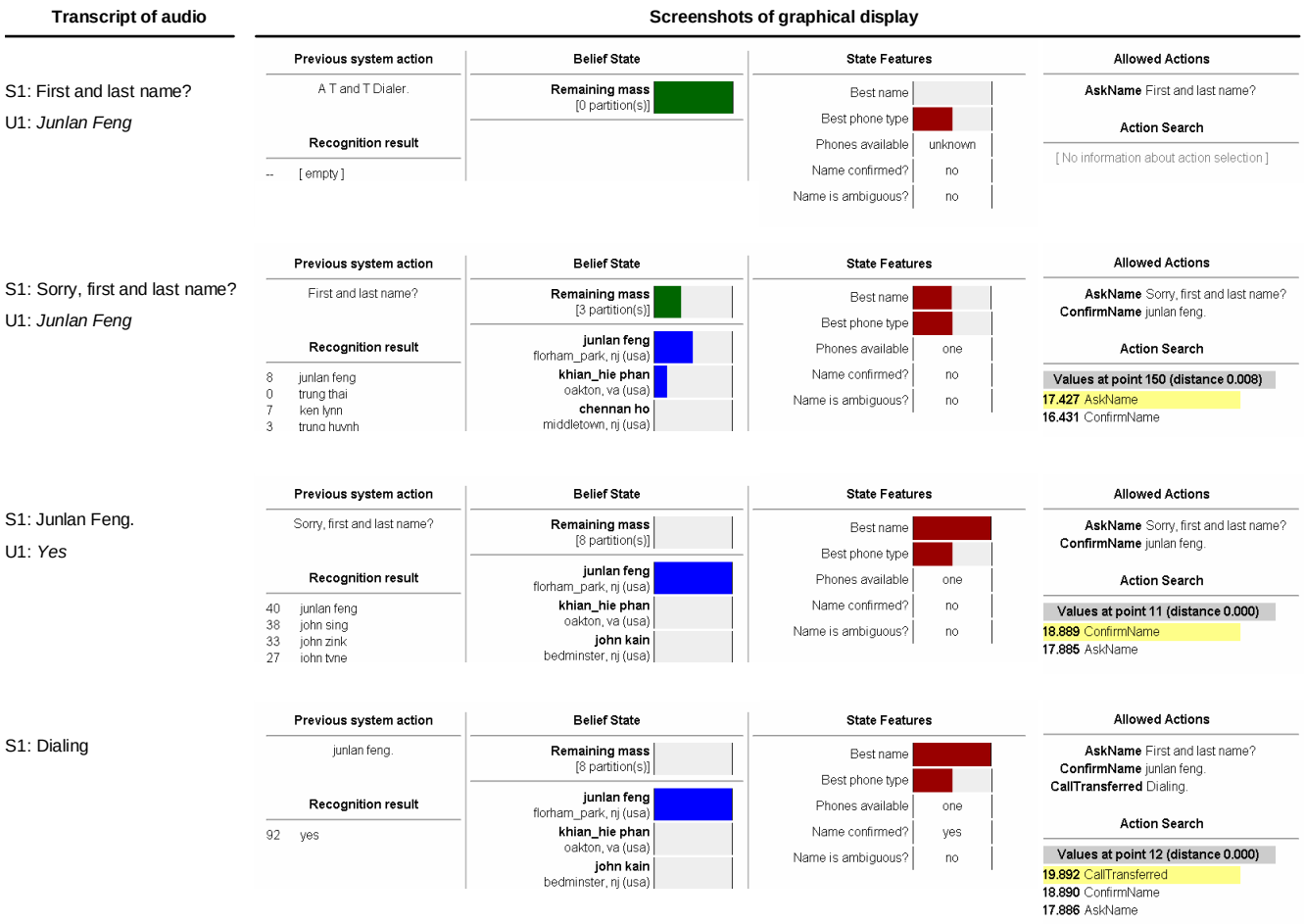


Figure 2: *The demonstration's graphical display during a call. The graphical display has been cropped and re-arranged for readability. The caller says "Junlan Feng" twice, and although each name recognition alone carries a low confidence score, the belief state aggregates this information. This novel behavior enables the call to progress faster than in the conventional system and illustrates one benefit of the POMDP approach. We have observed several other novel strategies not in a baseline conventional dialer: for example, the POMDP-based system will confirm a callee's name at different confidence levels depending on whether the callee has a phone number listed or not; and uses yes/no confirmation questions to disambiguate when there are two ambiguous callees.*

Generating research websites using summarisation techniques

Advaith Siddharthan & Ann Copestake
Natural Language and Information Processing Group
Computer Laboratory, University of Cambridge
{as372,aac10}@cl.cam.ac.uk

Abstract

We describe an application that generates web pages for research institutions by summarising terms extracted from individual researchers' publication titles. Our online demo covers all researchers and research groups in the Computer Laboratory, University of Cambridge. We also present a novel visualisation interface for browsing collaborations.

1 Introduction

Many research organisations organise their websites as a tree (e.g., department pages → research group pages → researcher pages). Individual researchers take responsibility for maintaining their own web pages and, in addition, researchers are organised into research groups that also maintain a web page. In this framework, information easily gets outdated, and publications lists generally stay more up-to-date than research summaries. Also, as individuals maintain their own web pages, connections between researchers in the organisation are often hard to find; a surfer then needs to move up and down the tree hierarchy to browse the profiles of different people. Browsing is also diffcult because individual web pages are organised differently, since standardised stylesheets are often considered inappropriate for diverse organisations.

Research summary pages using stylesheets can offer alternative methods of information access and browsing, aiding navigation and providing different views for different user needs, but these are time-consuming to create and maintain by hand. We are exploring the idea of automatically generated and updated web pages that accurately reflect the research interests being pursued within a research institution. We take as input existing personal pages from the Computer Laboratory, University of Cambridge, that contain publication lists in html. In our automatically generated pages, content (a research summary) is extracted from publication titles, and hence stays up-to-date provided individual researchers maintain their publication lists. Note that publication information is increasingly available through other sources, such as Google Scholar.

We aim to format information in a way that facilitates browsing; a screen shot is shown in Figure 1 for the researcher *Frank Stajano*, who is a member of the *Security* and *DTG* research groups. The left of the page contains links to researchers of the same research groups and the middle contains a research profile in the form of lists of key phrases presented in five year intervals (by publication date). In addition, the right of the page contains a list of recommendations: other researchers with similar research interests. Web pages for research groups are created by summarising the research profiles of individual members. In addition, we present a novel interactive visualisation that we have developed for displaying collaborations with the rest of the world.

In this paper we describe our methodology for identifying terms, clustering them and then creating research summaries (§2) and a generative summariser of collaborations (§4) that plugs into a novel visualisation (§3). An online demo is available at:
http://www.cl.cam.ac.uk/research/nl/webpage-demo/NLIP.html

2 Summarising research output

Our program starts with a list of publications extracted from researcher web pages; for example:

- S. Teufel. 2007. *An Overview of evaluation methods in TREC Ad-hoc Information Retrieval and TREC Question Answering.* In Evaluation of Text and Speech Systems. L. Dybkjaer, H. Hemsen, W. Minker (Eds.) Springer, Dordrecht (The Netherlands).

Proceedings of the ACL-08: HLT Demo Session (Companion Volume), pages 5–8,
Columbus, June 2008.

From each publication entry such as that above, the program extracts *author names*, *title* and *year of publication*. This is the only information used. We do not use the full paper, as pdfs are not available for all papers in publication pages (due to copyright and other issues). The titles are then parsed using the RASP parser (Briscoe and Carroll, 2002) and key-phrases are extracted by pattern matching. From the publication entry above, the extracted title:

> "An overview of evaluation methods in TREC ad-hoc information retrieval and TREC question answering"

produces five key-phrases:

> 'evaluation methods', 'evaluation methods in TREC ad-hoc information retrieval', 'TREC ad-hoc information retrieval', 'TREC question answering', 'information retrieval'

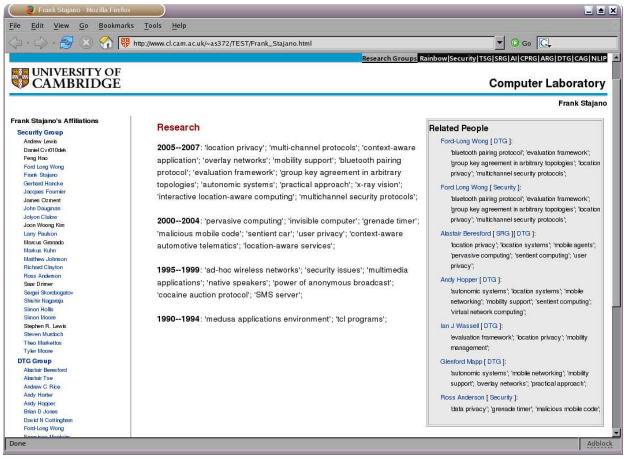


Figure 1: Screenshot: researcher web page.
http://www.cl.cam.ac.uk/research/nl/webpage-demo/Frank_Stajano.html

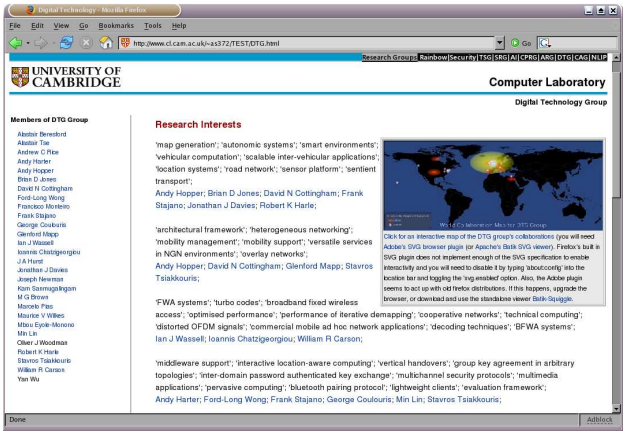


Figure 2: Screenshot: research group web page.
http://www.cl.cam.ac.uk/research/nl/webpage-demo/DTG.html

2.1 Individual researcher summaries

To create a web page for an individual researcher, the key-phrases extracted from all the paper titles authored by that researcher are clustered together based on similarity - an example cluster is shown below (from Karen Sparck Jones' profile):

> 'automatic classification for information retrieval', 'intelligent automatic information retrieval', 'information retrieval test collections', 'information retrieval system', 'automatic classification', 'intelligent retrieval', 'information retrieval', 'information science', 'test collections', 'mail retrieval', 'trec ad-hoc information retrieval'

A representative phrase (most similar to others in the cluster) is selected from each cluster ('*information retrieval*' from the above) and this phrase is linked with all the publication dates for papers the terms in the cluster come from. These extracted key-phrases are enumerated as lists in five year intervals; for example (from Karen Sparck Jones' profile):

> **1990–1994:** 'information retrieval'; 'document retrieval'; 'video mail retrieval'; 'automatic summarisation'; 'belief revision'; 'discourse structure'; 'cambridge/olivetti retrieval system'; 'system architecture'; 'agent interaction'; 'better NLP system evaluation'; 'early classification work'; 'text retrieval'; 'discourse modelling'...;

2.2 Recommendations (related people)

Recommendations for related people are generated by comparing the terms extracted between 2000 and 2008 for each researcher in the Computer Laboratory. The (at most) seven most similar researchers are shown in tabular form along with a list of terms from their profiles that are relevant to the researcher being viewed. These term lists inform the user as to why they might find the related people relevant.

2.3 Research Group Pages

Group pages are produced by summarising the pages of members of the group. Terms from individual research profiles are clustered according to who is working on them (gleaned from the author lists of the the associated paper title). The group page is presented as a list of clusters. This presentation shows how group members collaborate, and for each term shows the relevant researchers, making navigation

easier. Two clusters for the Graphics and Interaction (Rainbow) Group are show below to illustrate:

> 'histogram warping'; 'non-uniform b-spline subdivision'; 'stylised rendering'; 'multiresolution image representation'; 'human behaviour'; 'subdivision schemes'; 'minimising gaussian curvature variation near extraordinary vertices'; 'sampled cp surfaces'; 'bounded curvature variants': **Neil Dodgson; Thomas Cashman; Ursula Augsdorfer;**

> 'text for multiprojector tiled displays'; 'tabletop interface'; 'high-resolution tabletop applications'; 'distributed tabletops'; 'remote review meetings'; 'rapid prototyping': **Peter Robinson; Philip Tuddenham;**

3 Visualisation

Scalable Vector Graphics (SVG)[1] is a language for describing two-dimensional graphics and graphical applications in XML. Interactive images such as those in Figure 3 are produced by an XSLT script that transforms an input XML data file containing information about collaborations and latitudes and longitudes of cities and countries into an SVG representation[2]. This can be viewed through an Adobe Browser Plugin[3]. In the map, circles indicate the locations of co-authors of members of the NLIP research group, their size being proportional to the number of co-authors at that location. The map can be zoomed into, and at sufficient zoom, place names are made visible. Clicking on a location (circle) provides a summary of the collaboration (the summarisation is described in §4), while clicking on a country (oval) provides a contrywise overview such as:

> In the Netherlands, the NLIP Group has collaborators in Philips Research (Eindhoven), University of Twente (Enschede), Vrije Universiteit (VU) (Amsterdam) and University of Nijmegen.

4 Summarising collaborations

Our summarisation module slots into the visualisation interface; an example is shown in Figure 4. The aim is to summarise the topics that members of the research group collaborate with the researchers in

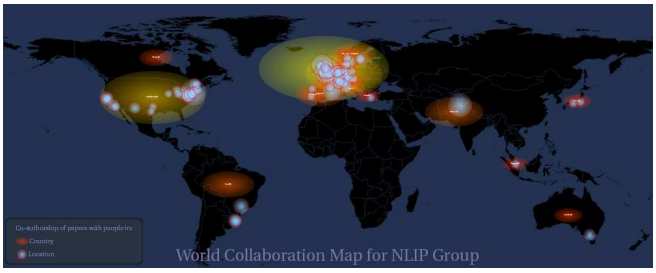


Figure 3: Screenshot: Visualisation of Collaboration between the NLIP Group and the rest of the world

Figure 4: Screenshot: Visualisation of Collaborations of ARG Group; zoomed into Europe and having clicked on Catonia (Italy) for a popup summary

each location on. The space constraints are dictated by the interface. To keep the visualisation clean, we enforce a four sentence limit for the summaries. There are four elements that each sentence contains— names of researchers in research group, names of researchers at location, terms that summarise the collaboration, and years of collaboration.

Our summaries are produced by an iterative process of clustering and summarising. In the first step, terms (key phrases) are extracted from all the papers that have co-authors in the location. Each term is tagged with the year(s) of publication and the names of researchers involved. These terms are then clustered based on the similarity of words in the terms and the similarity of their authors. Each such cluster contributes one sentence to the summary. The clustering process is pragmatic; the four sentence per summary limit means that at most four clusters should be formed. This means coarser clustering (fewer and larger clusters) for locations with many collaborations and finer-grained (more and smaller clusters) for locations with fewer collaborations.

The next step is to generate a sentence from each cluster. In this step, the terms in a sentence cluster are reclustered according to their date tag. then each time period is realised separately within the sentence, for example:

[1] http://www.w3.org/Graphics/SVG/

[2] Author Affiliations and Latitudes/Longitudes are semi-automatically extracted from the internet and hand corrected. The visualisation is only available for some research groups.

[3] http://www.adobe.com/svg/viewer/install/main.html

> Lawrence C Paulson collaborated with Cristiano Longo and Giampaolo Bella from 1997 to 2003 on 'formal verification', 'industrial payment and non-repudiation protocol', 'kerberos authentication system' and 'secrecy goals' and in 2006 on 'cardholder registration in Set' and 'accountability protocols'.

To make the summaries more readable, lists of conjunctions are restricted to a maximum length of four. Terms are incorporated into the list in decreasing order of frequency of occurrence. Splitting the sentence above into two time periods allows for the inclusion of more terms, without violating the restriction on list length. This form of sentence splitting is also pragmatic and is performed more aggressively in summaries with fewer sentences, having the effect of making short summaries slightly longer. Another method for increasing the number of terms is by aggregating similar terms. In the example below, three terms (*video mail retrieval*, *information retrieval* and *document retrieval*) are aggregated into one term. Thus six terms have made it to the clause, while keeping to the four terms per list limit.

> In the mid 1990s, K Sparck Jones, S J Young and M G Brown collaborated with J T Foote on 'video mail, information and document retrieval', 'cambridge/olivetti retrieval system', 'multimedia documents' and 'broadcast news'.

The four word limit is also enforced on lists of people. If there are too many people, the program refers to them by affiliation; for example:

> Joe Hurd collaborated with University of Utah on 'theorem proving', 'encryption algorithms', 'functional correctness proofs' and 'Arm verification'.

5 Discussion and Conclusions

Our summarisation strategy mirrors the multi-document summarisation strategy of Barzilay (2003), where sentences in the input documents are clustered according to their similarity. Larger clusters represent information that is repeated more often; hence the size of a cluster is indicative of importance. The novelty of our application is that this strategy has been used at a sub-sentential level, to summarise terms that are then used to generate sentences. While there has been research on generative summarisation, much of this has been focused on sentence extraction followed by some rewrite operation (e.g., sentence shortening (Vanderwende et al., 2007; Zajic et al., 2006; Conroy et al., 2004), aggregation (Barzilay, 2003) or reference regeneration (Siddharthan et al., 2004; Nenkova and McKeown, 2003)). In contrast, our system does not extract sentences at all; rather, it extracts terms from paper titles and our summaries are produced by clustering, summarising, aggregating and generalising over sets of terms and people. Our space constraints are dictated by by our visualisation interface, and our program employs pragmatic clustering and generalisation based on the amount of information it needs to summarise.

Acknowledgements

This work was funded by the Computer Laboratory, University of Cambridge, and the EPSRC (EP/C010035/1 and EP/F012950/1).

References

R. Barzilay. 2003. *Information Fusion for Multidocument Summarization: Paraphrasing & Generation*. Ph.D. thesis, Columbia University.

E.J. Briscoe and J. Carroll. 2002. Robust accurate statistical annotation of general text. In *Proceedings of the 3rd International Conference on Language Resources and Evaluation*, pages 1499–1504, Las Palmas, Gran Canaria.

J.M. Conroy, J.D. Schlesinger, J. Goldstein, and D.P. O'Leary. 2004. Left-brain/right-brain multi-document summarization. *Proceedings of DUC 2004*.

A. Nenkova and K. McKeown. 2003. References to named entities: a corpus study. *Companion proceedings of HLT-NAACL 2003–short papers-Volume 2*, pages 70–72.

A. Siddharthan, A. Nenkova, and K. McKeown. 2004. Syntactic simplification for improving content selection in multi-document summarization. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING 2004)*, pages 896–902, Geneva, Switzerland.

L. Vanderwende, H. Suzuki, C. Brockett, and A. Nenkova. 2007. Beyond SumBasic: Task-focused summarization with sentence simplification and lexical expansion. *Information Processing and Management*, 43(6):1606–1618.

D. Zajic, B. Dorr, J. Lin, and R. Schwartz. 2006. Sentence Compression as a Component of a Multi-Document Summarization System. *Proceedings of the 2006 Document Understanding Workshop, New York*.

BART: A Modular Toolkit for Coreference Resolution

Yannick Versley
University of Tübingen
versley@sfs.uni-tuebingen.de

Simone Paolo Ponzetto
EML Research gGmbH
ponzetto@eml-research.de

Massimo Poesio
University of Essex
poesio@essex.ac.uk

Vladimir Eidelman
Columbia University
vae2101@columbia.edu

Alan Jern
UCLA
ajern@ucla.edu

Jason Smith
Johns Hopkins University
jsmith@jhu.edu

Xiaofeng Yang
Inst. for Infocomm Research
xiaofengy@i2r.a-star.edu.sg

Alessandro Moschitti
University of Trento
moschitti@dit.unitn.it

Abstract

Developing a full coreference system able to run all the way from raw text to semantic interpretation is a considerable engineering effort, yet there is very limited availability of off-the shelf tools for researchers whose interests are not in coreference, or for researchers who want to concentrate on a specific aspect of the problem. We present BART, a highly modular toolkit for developing coreference applications. In the Johns Hopkins workshop on using lexical and encyclopedic knowledge for entity disambiguation, the toolkit was used to extend a reimplementation of the Soon et al. (2001) proposal with a variety of additional syntactic and knowledge-based features, and experiment with alternative resolution processes, preprocessing tools, and classifiers.

1 Introduction

Coreference resolution refers to the task of identifying noun phrases that refer to the same extralinguistic entity in a text. Using coreference information has been shown to be beneficial in a number of other tasks, including information extraction (McCarthy and Lehnert, 1995), question answering (Morton, 2000) and summarization (Steinberger et al., 2007). Developing a full coreference system, however, is a considerable engineering effort, which is why a large body of research concerned with feature engineering or learning methods (e.g. Culotta et al. 2007; Denis and Baldridge 2007) uses a simpler but non-realistic setting, using pre-identified mentions, and the use of coreference information in summarization or question answering techniques is not as widespread as it could be. We believe that the availability of a modular toolkit for coreference will significantly lower the entrance barrier for researchers interested in coreference resolution, as well as provide a component that can be easily integrated into other NLP applications.

A number of systems that perform coreference resolution are publicly available, such as GUITAR (Steinberger et al., 2007), which handles the full coreference task, and JAVARAP (Qiu et al., 2004), which only resolves pronouns. However, literature on coreference resolution, if providing a baseline, usually uses the algorithm and feature set of Soon et al. (2001) for this purpose.

Using the built-in maximum entropy learner with feature combination, BART reaches 65.8% F-measure on MUC6 and 62.9% F-measure on MUC7 using Soon et al.'s features, outperforming JAVARAP on pronoun resolution, as well as the Soon et al. reimplementation of Uryupina (2006). Using a specialized tagger for ACE mentions and an extended feature set including syntactic features (e.g. using tree kernels to represent the syntactic relation between anaphor and antecedent, cf. Yang et al. 2006), as well as features based on knowledge extracted from Wikipedia (cf. Ponzetto and Smith, in preparation), BART reaches state-of-the-art results on ACE-2. Table 1 compares our results, obtained using this extended feature set, with results from Ng (2007). Pronoun resolution using the extended feature set gives 73.4% recall, coming near specialized pronoun resolution systems such as (Denis and Baldridge, 2007).

Proceedings of the ACL-08: HLT Demo Session (Companion Volume), pages 9–12,
Columbus, June 2008.

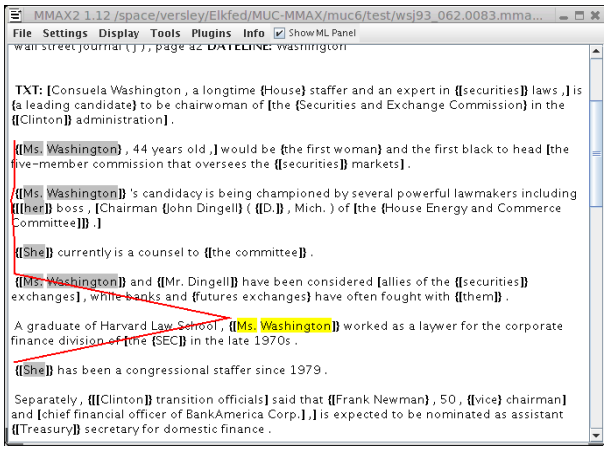


Figure 1: Results analysis in MMAX2

2 System Architecture

The BART toolkit has been developed as a tool to explore the integration of knowledge-rich features into a coreference system at the Johns Hopkins Summer Workshop 2007. It is based on code and ideas from the system of Ponzetto and Strube (2006), but also includes some ideas from GUITAR (Steinberger et al., 2007) and other coreference systems (Versley, 2006; Yang et al., 2006). [1]

The goal of bringing together state-of-the-art approaches to different aspects of coreference resolution, including specialized preprocessing and syntax-based features has led to a design that is very modular. This design provides effective separation of concerns across several several tasks/roles, including engineering *new features* that exploit different sources of knowledge, designing improved or specialized *preprocessing* methods, and improving the way that coreference resolution is mapped to a *machine learning* problem.

Preprocessing To store results of preprocessing components, BART uses the standoff format of the MMAX2 annotation tool (Müller and Strube, 2006) with MiniDiscourse, a library that efficiently implements a subset of MMAX2's functions. Using a generic format for standoff annotation allows the use of the coreference resolution as part of a larger system, but also performing qualitative error analysis using integrated MMAX2 functionality (annotation diff, visual display).

Preprocessing consists in marking up noun chunks and named entities, as well as additional information such as part-of-speech tags and merging these information into markables that are the starting point for the mentions used by the coreference resolution proper.

Starting out with a **chunking pipeline**, which uses a classical combination of tagger and chunker, with the Stanford POS tagger (Toutanova et al., 2003), the YamCha chunker (Kudoh and Matsumoto, 2000) and the Stanford Named Entity Recognizer (Finkel et al., 2005), the desire to use richer syntactic representations led to the development of a **parsing pipeline**, which uses Charniak and Johnson's reranking parser (Charniak and Johnson, 2005) to assign POS tags and uses base NPs as chunk equivalents, while also providing syntactic trees that can be used by feature extractors. BART also supports using the Berkeley parser (Petrov et al., 2006), yielding an easy-to-use Java-only solution.

To provide a better starting point for mention detection on the ACE corpora, the **Carafe pipeline** uses an ACE mention tagger provided by MITRE (Wellner and Vilain, 2006). A specialized merger then discards any base NP that was not detected to be an ACE mention.

To perform coreference resolution proper, the mention-building module uses the markables created by the pipeline to create mention objects, which provide an interface more appropriate for coreference resolution than the MiniDiscourse markables. These objects are grouped into equivalence classes by the resolution process and a coreference layer is written into the document, which can be used for detailed error analysis.

Feature Extraction BART's default resolver goes through all mentions and looks for possible antecedents in previous mentions as described by Soon et al. (2001). Each pair of anaphor and candidate is represented as a `PairInstance` object, which is enriched with classification features by feature extractors, and then handed over to a machine learning-based classifier that decides, given the features, whether anaphor and candidate are coreferent or not. Feature extractors are realized as separate classes, allowing for their independent develop-

[1] An open source version of BART is available from `http://www.sfs.uni-tuebingen.de/~versley/BART/`.

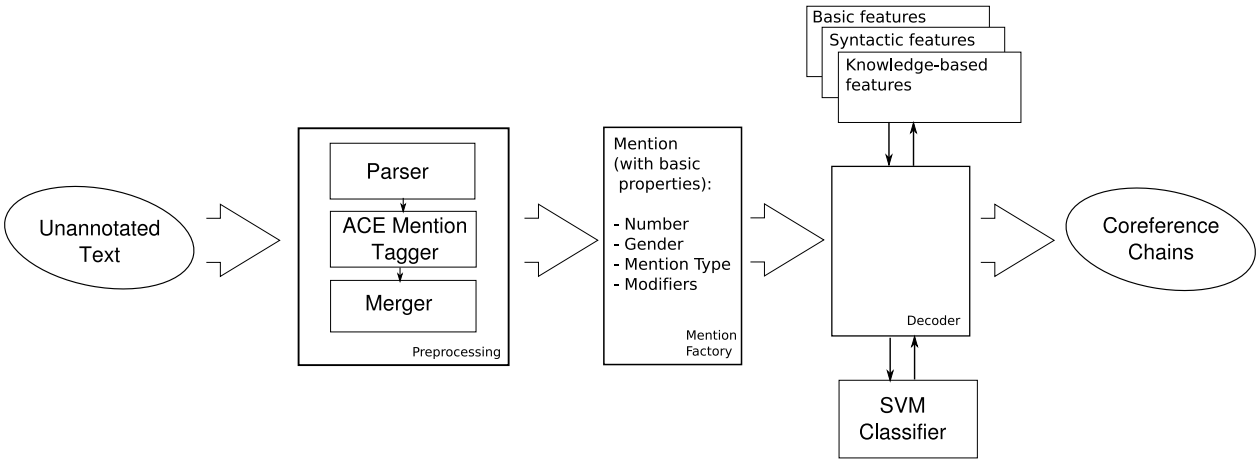


Figure 2: Example system configuration

ment. The set of feature extractors that the system uses is set in an XML description file, which allows for straightforward prototyping and experimentation with different feature sets.

Learning BART provides a generic abstraction layer that maps application-internal representations to a suitable format for several machine learning toolkits: One module exposes the functionality of the the **WEKA** machine learning toolkit (Witten and Frank, 2005), while others interface to specialized state-of-the art learners. **SVMLight** (Joachims, 1999), in the SVMLight/TK (Moschitti, 2006) variant, allows to use tree-valued features. SVM Classification uses a Java Native Interface-based wrapper replacing SVMLight/TK's `svm_classify` program to improve the classification speed. Also included is a **Maximum entropy** classifier that is based upon Robert Dodier's translation of Liu and Nocedal's (1989) L-BFGS optimization code, with a function for programmatic feature combination.[2]

Training/Testing The training and testing phases slightly differ from each other. In the training phase, the pairs that are to be used as training examples have to be selected in a process of sample selection, whereas in the testing phase, it has to be decided which pairs are to be given to the decision function and how to group mentions into equivalence relations given the classifier decisions.

This functionality is factored out into the *encoder/decoder* component, which is separate from feature extraction and machine learning itself. It is possible to completely change the basic behavior of the coreference system by providing new encoders/decoders, and still rely on the surrounding infrastructure for feature extraction and machine learning components.

3 Using BART

Although BART is primarily meant as a platform for experimentation, it can be used simply as a coreference resolver, with a performance close to state of the art. It is possible to import raw text, perform preprocessing and coreference resolution, and either work on the MMAX2-format files, or export the results to arbitrary inline XML formats using XSL stylesheets.

Adapting BART to a new coreferentially annotated corpus (which may have different rules for mention extraction – witness the differences between the annotation guidelines of MUC and ACE corpora) usually involves fine-tuning of mention creation (using pipeline and MentionFactory settings), as well as the selection and fine-tuning of classifier and features. While it is possible to make radical changes in the preprocessing by re-engineering complete pipeline components, it is usually possible to achieve the bulk of the task by simply mixing and matching existing components for preprocessing and feature extraction, which is possible by modifying only configuration settings and an XML-

[2] see `http://riso.sourceforge.net`

	BNews			NPaper			NWire		
	Recl	Prec	F	Recl	Prec	F	Recl	Prec	F
basic feature set	0.594	0.522	0.556	0.663	0.526	0.586	0.608	0.474	0.533
extended feature set	0.607	0.654	0.630	0.641	0.677	0.658	0.604	0.652	0.627
Ng 2007*	0.561	0.763	0.647	0.544	0.797	0.646	0.535	0.775	0.633

*: "expanded feature set" in Ng 2007; Ng trains on the entire ACE training corpus.

Table 1: Performance on ACE-2 corpora, basic vs. extended feature set

based description of the feature set and learner(s) used.

Several research groups focusing on coreference resolution, including two not involved in the initial creation of BART, are using it as a platform for research including the use of new information sources (which can be easily incorporated into the coreference resolution process as features), different resolution algorithms that aim at enhancing global coherence of coreference chains, and also adapting BART to different corpora. Through the availability of BART as open source, as well as its modularity and adaptability, we hope to create a larger community that allows both to push the state of the art further and to make these improvements available to users of coreference resolution.

Acknowledgements We thank the CLSP at Johns Hopkins, NSF and the Department of Defense for ensuring funding for the workshop and to EML Research, MITRE, the Center for Excellence in HLT, and FBK-IRST, that provided partial support. Yannick Versley was supported by the Deutsche Forschungsgesellschaft as part of SFB 441 "Linguistic Data Structures"; Simone Paolo Ponzetto has been supported by the Klaus Tschira Foundation (grant 09.003.2004).

References

Charniak, E. and Johnson, M. (2005). Coarse-to-fine n-best parsing and maxent discriminative reranking. In *Proc. ACL 2005*.

Culotta, A., Wick, M., and McCallum, A. (2007). First-order probabilistic models for coreference resolution. In *Proc. HLT/NAACL 2007*.

Denis, P. and Baldridge, J. (2007). A ranking approach to pronoun resolution. In *Proc. IJCAI 2007*.

Finkel, J. R., Grenager, T., and Manning, C. (2005). Incorporating non-local information into information extraction systems by Gibbs sampling. In *Proc. ACL 2005*, pages 363–370.

Joachims, T. (1999). Making large-scale SVM learning practical. In Schölkopf, B., Burges, C., and Smola, A., editors, *Advances in Kernel Methods - Support Vector Learning*.

Kudoh, T. and Matsumoto, Y. (2000). Use of Support Vector Machines for chunk identification. In *Proc. CoNLL 2000*.

Liu, D. C. and Nocedal, J. (1989). On the limited memory method for large scale optimization. *Mathematical Programming B*, 45(3):503–528.

McCarthy, J. F. and Lehnert, W. G. (1995). Using decision trees for coreference resolution. In *Proc. IJCAI 1995*.

Morton, T. S. (2000). Coreference for NLP applications. In *Proc. ACL 2000*.

Moschitti, A. (2006). Making tree kernels practical for natural language learning. In *Proc. EACL 2006*.

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

Ng, V. (2007). Shallow semantics for coreference resolution. In *Proc. IJCAI 2007*.

Petrov, S., Barett, L., Thibaux, R., and Klein, D. (2006). Learning accurate, compact, and interpretable tree annotation. In *COLING-ACL 2006*.

Ponzetto, S. P. and Strube, M. (2006). Exploiting semantic role labeling, WordNet and Wikipedia for coreference resolution. In *Proc. HLT/NAACL 2006*.

Qiu, L., Kan, M.-Y., and Chua, T.-S. (2004). A public reference implementation of the RAP anaphora resolution algorithm. In *Proc. LREC 2004*.

Soon, W. M., Ng, H. T., and Lim, D. C. Y. (2001). A machine learning approach to coreference resolution of noun phrases. *Computational Linguistics*, 27(4):521–544.

Steinberger, J., Poesio, M., Kabadjov, M., and Jezek, K. (2007). Two uses of anaphora resolution in summarization. *Information Processing and Management*, 43:1663–1680. Special issue on Summarization.

Toutanova, K., Klein, D., Manning, C. D., and Singer, Y. (2003). Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proc. NAACL 2003*, pages 252–259.

Uryupina, O. (2006). Coreference resolution with and without linguistic knowledge. In *Proc. LREC 2006*.

Versley, Y. (2006). A constraint-based approach to noun phrase coreference resolution in German newspaper text. In *Konferenz zur Verarbeitung Natürlicher Sprache (KONVENS 2006)*.

Wellner, B. and Vilain, M. (2006). Leveraging machine readable dictionaries in discriminative sequence models. In *Proc. LREC 2006*.

Witten, I. and Frank, E. (2005). *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann.

Yang, X., Su, J., and Tan, C. L. (2006). Kernel-based pronoun resolution with structured syntactic knowledge. In *Proc. CoLing/ACL-2006*.

Demonstration of the UAM CorpusTool for text and image annotation

Mick O'Donnell
Escuela Politécnica Superior
Universidad Autónoma de Madrid
28049, Cantoblanco, Madrid, Spain
michael.odonnell@uam.es

Abstract

This paper introduced the main features of the UAM CorpusTool, software for human and semi-automatic annotation of text and images. The demonstration will show how to set up an annotation project, how to annotate text files at multiple annotation levels, how to automatically assign tags to segments matching lexical patterns, and how to perform cross-layer searches of the corpus.

1 Introduction

In the last 20 years, a number of tools have been developed to facilitate the human annotation of text. These have been necessary where software for automatic annotation has not been available, e.g., for linguistic patterns which are not easily identified by machine, or for languages without sufficient linguistic resources.

The vast majority of these annotation tools have been developed for particular projects, and have thus not been readily adaptable to different annotation problems. Often, the annotation scheme has been built into the software, or the software has been limited in that they allow only certain types of annotation to take place.

A small number of systems have however been developed to be general purpose text annotation systems, e.g., MMAX-2 (Müller and Strube 2006), GATE (Cunningham et al 2002), WordFreak (Morton and LaCivita 2003) and Knowtator (Ogren 2006).

With the exception of the last of these however, these systems are generally aimed at technically advanced users. WordFreak, for instance, requires writing of Java code to adapt to a different annotation scheme. Users of MMAX-2 need to edit XML by hand to provide annotation schemes. Gate allows editing of annotation schemes within the tool, but it is a very complex system, and lacks clear documentation to help the novice user become competent.

The UAM CorpusTool is a text annotation tool primarily aimed at the linguist or computational linguist who does not program, and would rather spend their time annotating text than learning how to use the system. The software is thus designed from the ground up to support typical user workflow, and everything the user needs to perform annotation tasks is included within the software.

2 The Project Window

In the majority of cases, the annotator is interested in annotating a range of texts, not just single texts. Additionally, in most cases annotation at multiple linguistic levels is desired (e.g., classifying the text as a whole, tagging sections of text by function (e.g., abstract, introduction, etc.), tagging sentences/clauses, and tagging participants in clauses. To overcome the complexity of dealing with multiple source files annotated at multiple levels, the main window of the CorpusTool is thus a window for project management (see Figure 1).

Proceedings of the ACL-08: HLT Demo Session (Companion Volume), pages 13–16,
Columbus, June 2008.

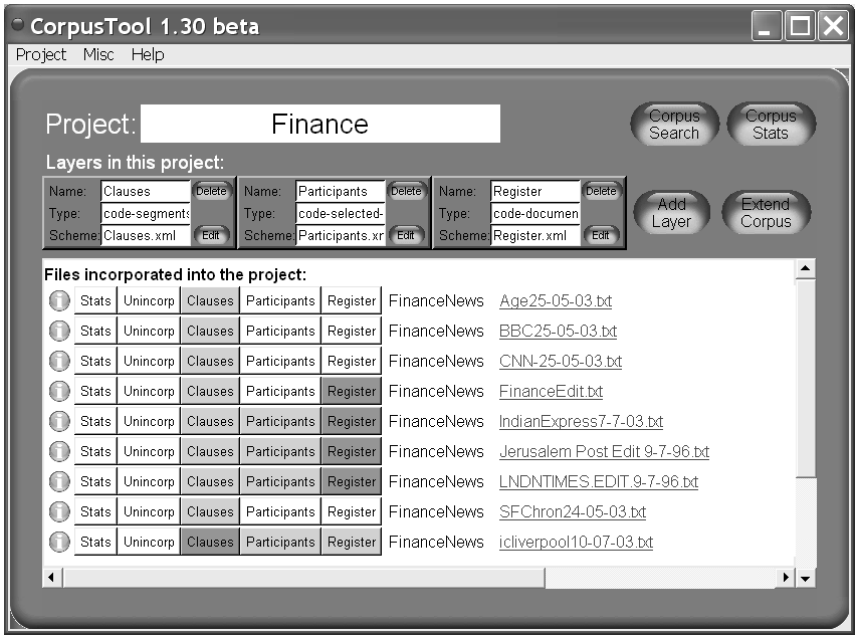


Figure 1: The Project Window of UAM CorpusTool

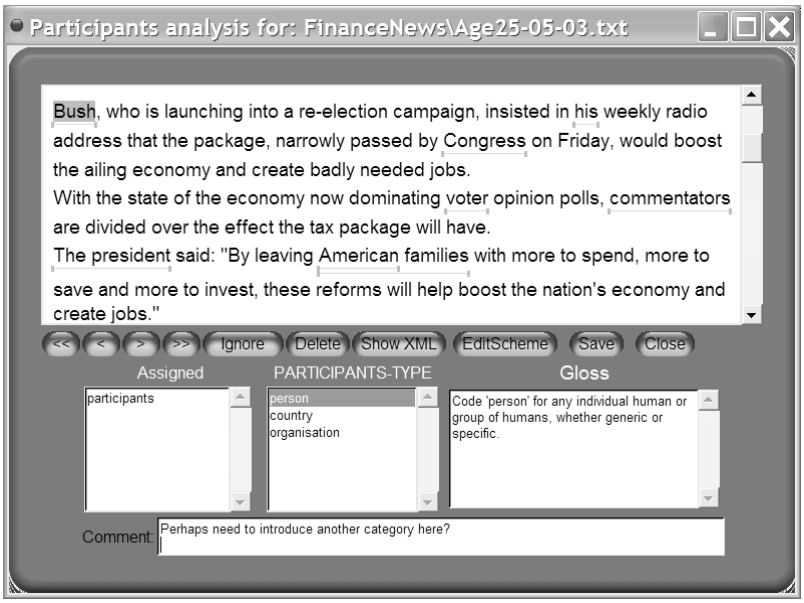


Figure 3: An annotation window for 'Participant' layer.

```
<?xml version='1.0' encoding='utf-8'?>
<document>
  <segments>
    <segment id='1' start='158' end='176'
             features='participant;human' state='active'/>
    <segment id='2' start='207' end='214'
             features='participant;organisation;company'
             state='active'/>
        ...
  </segments>
</document>
```

Figure 4: Annotation Storage Example

This window allows the user to add new annotation layers to the project, and edit/extend the annotation scheme for each layer (by clicking on the "edit" button shown with each layer panel). It also allows the user to add or delete source files to the project, and to open a specific file for annotation at a specific layer (each file has a button for each layer).

3 Tag Hierarchy Editing

Most of the current text annotation tools lack built-in facilities for creating and editing the coding scheme (the tag set). UAM CorpusTool uses a hierarchally organised tag scheme, allowing cross-classification and multiple inheritance (both disjunctive and conjunctive). The scheme is edited graphically, adding, renaming, moving or deleting features, adding new sub-distinctions, etc. See Figure 3.

An important feature of the tool is that any change to the coding scheme is automatically propagated throughout all files annotated at this layer. For instance, if a feature is renamed in the scheme editor, it is also renamed in all annotation files.

The user can also associate a gloss with each tag, and during annotation, the gloss associated with each feature can be viewed to help the coder determine which tag to assign.

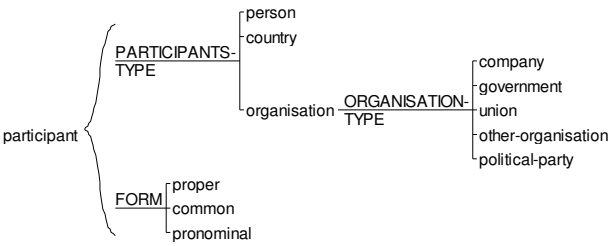


Figure 2: Graphical Editing of the Tag Hierarchy

4 Annotation Windows

When the user clicks on the button for a given text file/layer, an annotation window opens (see Figure 3). This window shows the text in the top panel (with previously identified text segments indicated with underlining). When the user creates a new segment (by swiping text) or selects an existing segment, the space below the text window shows controls to select the tags to assign to this segment. Tags are drawn from the tag scheme for the current layer. Since the tag hierarchy allows cross-classification, multiple tags are assigned to the segment. CorpusTool allows for partially overlapping segments, and embedding of segments.

Annotated texts are stored using stand-off XML, one file per source text and layer. See Figure 4 for a sample. The software does not currently input from or export to any of the various text encoding standards, but will be extended to do so as it becomes clear which standards users want supported.

Currently the tool only supports assigning tags to text. Annotating structural relations between text segments (e.g., co-reference, constituency or rhetorical relations) is not currently supported, but is planned for later releases.

5 Corpus Search

A button on the main window opens a Corpus Search interface, which allows users to retrieve lists of segments matching a query. Queries can involve multiple layers, for instance, `subject in passive-clause in english` would retrieve all NPs tagged as subject in clauses tagged as passive-clause in texts tagged as 'english' (this is thus a search over 3 annotation layers). Searches can also retrieve segments "containing" segments. One can also search for segments containing a string.

Where a lexicon is provided (currently only English), users can search for segments containing lexical patterns, for instance, `clause containing 'be% @participle'` would return all clause segments containing any inflection of 'be' immediately followed by any participle verb (i.e. most of the passive clauses). Since dictionaries are used, the text does not need to be pre-tagged with a POS tagger, which may be unreliable on texts of a different nature to those on which the tagger was trained. Results are displayed in a KWIK table format.

6 Automating Annotation

Currently, automatic segmentation into sentences is provided. I am currently working on automatic NP segmentation.

The search facility outlined above can also be used for semi-automatic tagging of text. To autocode segments as 'passive-clause', one specifies a search pattern (i.e., `clause containing`

'be% @participle'). The user is presented with all matches, with a check-box next to each. The user can then uncheck the hits which are false matches, and then click on the "Store" button to tag all checked segments with the 'passive-clause' feature. A reasonable number of syntactic features can be identified in this way.

7 Statistical processing

The tool comes with a statistical analysis interface which allows for specified sub-sections of the corpora (e.g., 'finite-clause in english' vs. 'finite-clause in spanish') to be described or contrasted. Statistics can be of the text itself (e.g., lexical density, pronominal usage, word and segment length, etc.), or relate to the frequency of annotations. These statistics can also be exported in tab-delimited form for processing in more general statistical packages.

8 Intercoder Reliability Testing

Where several users have annotated files at the same layers, a separate tool is provided to compare each annotation document, showing only the differences between coders, and also indicating total coder agreement. The software can also produce a "consensus" version of the annotations, taking the most popular coding where 3 or more coders have coded the document. In this way, each coder can be compared to the consensus (n comparisons), rather than comparing the n! pairs of documents.

9 Annotating Images

The tool can also be used to annotate images instead of text files. In this context, one can swipe regions of the image to create a selection, and assign features to the selection. Since stand-off annotation is used for both text and image, much of the code-base is common between the two applications. The major differences are: i) a different annotation widget is used for text selection than for image selection; ii) segments in text are defined by a tuple: (startchar, endchar), while image segments are defined by a tuple of points ((startx,starty), (endx,endy)), and iii) search in images is restricted to tag searching, while text can be searched for strings and lexical patterns.

10 Conclusions

UAM CorpusTool is perhaps the most user-friendly of the annotation tools available, offering easy installation, an intuitive interface, yet powerful facilities for management of multiple documents annotated at multiple levels.

The main limitation of the tool is that it currently deals only with feature tagging. Future work will add structural tagging, including co-reference linking, rhetorical structuring and syntactic structuring.

The use of the tool is rapidly spreading: in the first 15 months of availability, the tool has been downloaded 1700 times, to 1100 distinct CPUs (with only minimal advertisement). It is being used for various text annotation projects throughout the world, but mostly by individual linguists performing linguistic studies.

UAM CorpusTool is free, available currently for Macintosh and Windows machines. It is not open source at present, delivered as a standalone executable. It is implemented in Python, using TKinter .

Acknowledgments

The development of UAM CorpusTool was partially funded by the Spanish Ministry of Education and Science (MEC) under grant number HUM2005-01728/FILO (the WOSLAC project).

References

C. Müller, and 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 (English Corpus Linguistics, Vol.3)*. Frankfurt: Peter Lang. 197-214.

H. Cunningham, D. Maynard, K. Bontcheva and V. Tablan. 2002. GATE: A Framework and Graphical Development Environment for Robust NLP Tools and Applications. *Proceedings of the 40th Meeting of the Association for Computational Linguistics (ACL'02)*. Philadelphia, July 2002.

T.S. Morton and J. LaCivita. 2003. WordFreak: An Open Tool for Linguistic Annotation. *Proceedings of HLT-NAACL*. 17-18.

P.V. Ogren 2006. Knowtator: a plug-in for creating training and evaluation data sets for biomedical natural language systems. *Proceedings of the 9th International Protégé Conference*. 73–76.

Interactive ASR Error Correction for Touchscreen Devices

David Huggins-Daines
Language Technologies Institute
Carnegie Mellon University
Pittsburgh, PA 15213, USA
dhuggins@cs.cmu.edu

Alexander I. Rudnicky
Language Technologies Institute
Carnegie Mellon University
Pittsburgh, PA 15213, USA
air@cs.cmu.edu

Abstract

We will demonstrate a novel graphical interface for correcting search errors in the output of a speech recognizer. This interface allows the user to visualize the word lattice by "pulling apart" regions of the hypothesis to reveal a cloud of words simlar to the "tag clouds" popular in many Web applications. This interface is potentially useful for dictation on portable touchscreen devices such as the Nokia N800 and other mobile Internet devices.

1 Introduction

For most people, dictating continuous speech is considerably faster than entering text using a keyboard or other manual input device. This is particularly true on mobile devices which typically have no hardware keyboard whatsoever, a 12-digit keypad, or at best a miniaturized keyboard unsuitable for touch typing.

However, the effective speed of text input using speech is significantly reduced by the fact that even the best speech recognition systems make errors. After accounting for error correction, the effective number of words per minute attainable with speech recognition drops to within the range attainable by an average typist (Moore, 2004). Moreover, on a mobile phone with predictive text entry, it has been shown that isolated word dictation is actually slower than using a 12-digit keypad for typing SMS messages (Karpov et al., 2006).

2 Description

It has been shown that multimodal error correction methods are much more effective than using speech alone (Lewis, 1999). Mobile devices are increasingly being equipped with touchscreens which lend themselves to gesture-based interaction methods.

Therefore, we propose an interactive method of visualizing and browsing the word lattice using gestures in order to correct speech recognition errors. The user is presented with the decoding result in a large font, either in a window on the desktop, or in a full-screen presentation on a touchscreen device. If the utterance is too long to fit on the screen, the user can scroll left and right using touch gestures. The initial interface is shown in Figure 1.

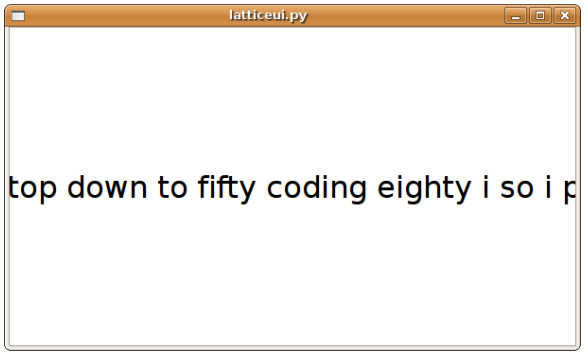


Figure 1: Initial hypothesis view

Where there is an error, the user can "pull apart" the result using a touch stroke (or a multitouch gesture where supported), revealing a "cloud" of hypothesis words at that point in the utterance, as shown in Figure 2.

It is also possible to expand the time interval over which the cloud is calculated by dragging sideways, resulting in a view like that in Figure 3. The user can then select zero or more words to add to the hypothesis string in place of the errorful text which was "exploded", as shown in Figure 4.

Proceedings of the ACL-08: HLT Demo Session (Companion Volume), pages 17–19,
Columbus, June 2008.

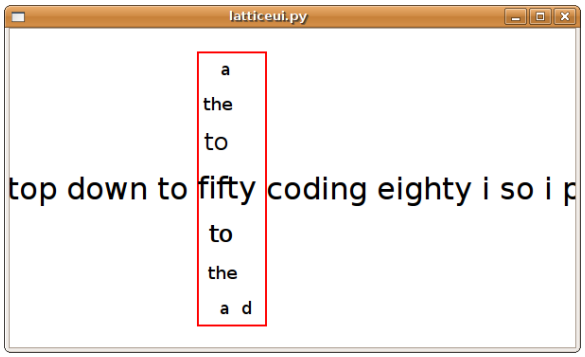


Figure 2: Expanded word view

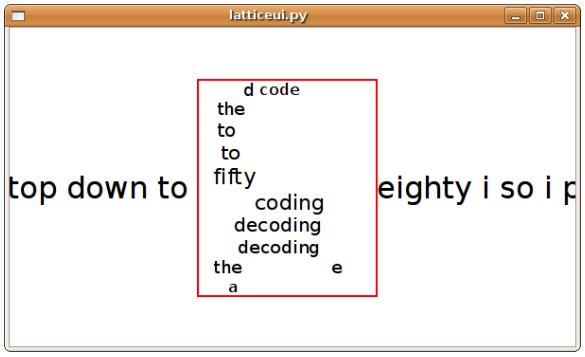


Figure 3: Word cloud expanded in time

The word cloud is constructed by finding all words active within a time interval whose log posterior probability falls within range of the most probable word. Word posterior probabilities are calculated using the forward-backward algorithm described in (Wessel et al., 1998). Specifically, given a word lattice in the form of a directed acyclic graph, whose nodes represent unique starting points t in time, and whose edges represent the acoustic likelihoods of word hypotheses w_s^t spanning a given time interval (s, t), we can calculate the forward variable $\alpha_t(w)$, which represents the joint probability of all word sequences ending in w_s^t and the acoustic observations up to time t, as:

$$\alpha_t(w) = P(O_1^s, w_s^t) = \sum_{v_s^t \in prev(w)} P(w|v)P(w_s^t)\alpha_s(v)$$

Here, $P(w|v)$ is the bigram probability of (v, w) obtained from the language model and $P(w_s^t)$ is the acoustic likelihood of the word model w given the observed speech from time s to t, as approximated by the Viterbi path score.

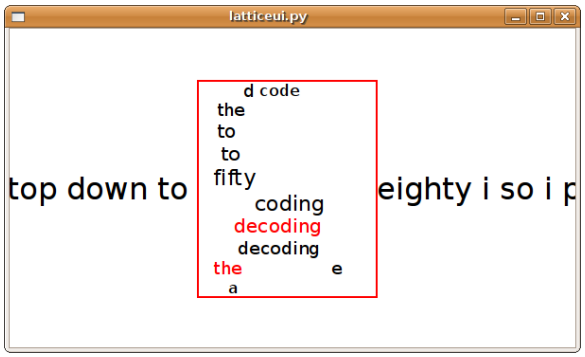


Figure 4: Selecting replacement words

Likewise, we can compute the backward variable $\beta_t(w)$, which represents the conditional probability of all word sequences beginning in w_s^t and the acoustic observations from time $t + 1$ to the end of the utterance, given w_s^t:

$$\beta_t(w) = P(O_t^T|w_s^t) = \sum_{v_t^e \in succ(w)} P(v|w)P(v_t^e)\beta_e(v)$$

The posterior probability $P(w_s^t|O_1^T)$ can then be obtained by multiplication and normalization:

$$P(w_s^t|O_1^T) = \frac{P(w_s^t, O_1^T)}{P(O_1^T)} = \frac{\alpha_t(w)\beta_t(w)}{P(O_1^T)}$$

This algorithm has a straightforward extension to trigram language models which has been omitted here for simplicity.

This interface is inspired by the web browser zooming interface used on the Apple iPhone (Apple, Inc., 2008), as well as the Speech Dasher lattice correction tool (Vertanen, 2004). We feel that it is potentially useful not only for automatic speech recognition, but also for machine translation and any other situation in which a lattice representation of a possibly errorful hypothesis is available. A video of this interface in Ogg Theora format[1] can be viewed at http://www.cs.cmu.edu/~dhuggins/touchcorrect.ogg.

[1]For Mac OS X: http://xiph.org/quicktime/download.html
For Windows: http://www.illiminable.com/ogg/downloads.html

3 Script Outline

For our demonstration, we will have available a poster describing the interaction method being demonstrated. We will begin by describing the motivation for this work, followed by a "silent" demo of the correction method itself, using pre-recorded audio. We will then demonstrate live speech input and correction using our own voices. The audience will then be invited to test the interaction method on a touchscreen device (either a handheld computer or a tablet PC).

4 Requirements

To present this demo, we will be bringing two Nokia Internet Tablets as well as a laptop and possibly a Tablet PC. We have no requirements from the conference organizers aside from a suitable number of power outlets, a table, and a poster board.

Acknowledgements

We wish to thank Nokia for donating an N800 Internet Tablet used to develop this software.

References

E. Karpov, I. Kiss, J. Leppänen, J. Olsen, D. Oria, S. Sivadas and J. Tian 2006. Short Message System dictation on Series 60 mobile phones. *Workshop on Speech in Mobile and Pervasive Environments (SiMPE) in Conjunction with MobileHCI 2006.* Helsinki, Finland.

Keith Vertanen 2004. *Efficient Computer Interfaces Using Continuous Gestures, Language Models, and Speech.* M.Phil Thesis, University of Cambridge, Cambridge, UK.

Apple, Inc. 2008. *iPhone: Zooming In to Enlarge Part of a Webpage.* http://docs.info.apple.com/article.html?artnum=305899

Roger K. Moore 2004. Modelling Data Entry Rates for ASR and Alternative Input Methods. *Proceedings of Interspeech 2004.* Jeju, Korea.

James R. Lewis 1999. Effect of Error Correction Strategy on Speech Dictation Throughput *Proceedings of the Human Factors and Ergonomics Society 43rd Annual Meeting.*

Frank Wessel, Klaus Macherey, Ralf Schlüter 1998. Using Word Probabilities as Confidence Measures. *Proceedings of ICASSP 1998.*

Yawat: Yet Another Word Alignment Tool

Ulrich Germann
University of Toronto
germann@cs.toronto.edu

Abstract

Yawat[1] is a tool for the visualization and manipulation of word- and phrase-level alignments of parallel text. Unlike most other tools for manual word alignment, it relies on dynamic markup to visualize alignment relations, that is, markup is shown and hidden depending on the current mouse position. This reduces the visual complexity of the visualization and allows the annotator to focus on one item at a time. For a bird's-eye view of alignment patterns within a sentence, the tool is also able to display alignments as alignment matrices. In addition, it allows for manual labeling of alignment relations with customizable tag sets. Different text colors are used to indicate which words in a given sentence pair have already been aligned, and which ones still need to be aligned. Tag sets and color schemes can easily be adapted to the needs of specific annotation projects through configuration files. The tool is implemented in JavaScript and designed to run as a web application.

1 Introduction

Sub-sentential alignments of parallel text play an important role in statistical machine translation (SMT). Aligning parallel data on the word- or phrase-level is typically one of the first steps in building SMT systems, as those alignments constitute the basis for the construction of probabilistic translation dictionaries. Consequently, considerable effort has gone into devising and improving automatic word alignment algorithms, and into evaluating their performance (e.g., Och and Ney, 2003; Taskar *et al.*, 2005; Moore *et al.*, 2006; Fraser and Marcu, 2006, among many others). For the sake of simplicity, we will in the following use the term "word alignment" to refer to any form of alignment that identifies words or groups of words as translations of each other.

Any explicit evaluation of word alignment quality requires human intervention at some point, be it in the direct evaluation of candidate word alignments produced by a word alignment system, or in the creation of a gold standard against which candidate word alignments can be compared automatically. This human intervention works best with an interactive, visual interface.

2 Word alignment visualization

Over the years, numerous tools for the visualization and creation of word alignments have been developed (e.g., Melamed, 1998; Smith and Jahr, 2000; Ahrenberg *et al.*, 2002; Rassier and Pedersen, 2003; Daumé; Tiedemann; Hwa and Madnani, 2004; Lambert, 2004; Tiedemann, 2006). Most of them employ one of two visualization techniques. The first is to draw lines between associated words, as shown in Fig. 1. The second is to use an alignment matrix (Fig. 2), where the rows of the matrix correspond to the words of the sentence in one language and the columns to the words of that sentence's translation into the other language. Marks in the matrix's cells indicate whether the words represented by the row and column of the cell are linked or not. A third technique, employed in addition to drawing lines by Melamed (1998) and as the sole mechanism by Tiedemann (2006), is to use colors to indicate which words correspond to each other on the two sides of the parallel corpus.

The three techniques just mentioned work reasonably well for very short sentences, but reach their limits quickly as sentence length increases. Alignment visualization by coloring schemes requires as many different colors as there are words in the (shorter) sentence. Alignment visualization by drawing lines and alignment matrices both require that each of the two sentences in each sentence pair is

[1] Yawat was first presented at the 2007 *Linguistic Annotation Workshop* (Germann, 2007).

Proceedings of the ACL-08: HLT Demo Session (Companion Volume), pages 20–23,
Columbus, June 2008.

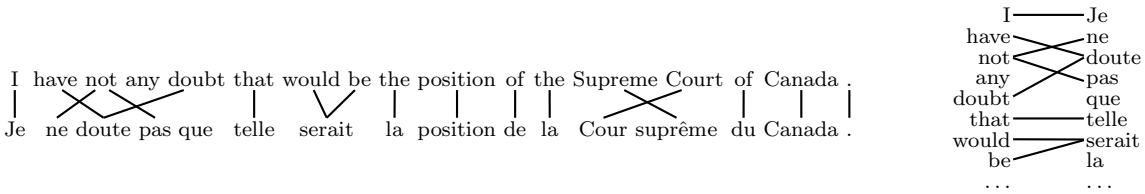


Figure 1: Visualization of word alignments by drawing lines.

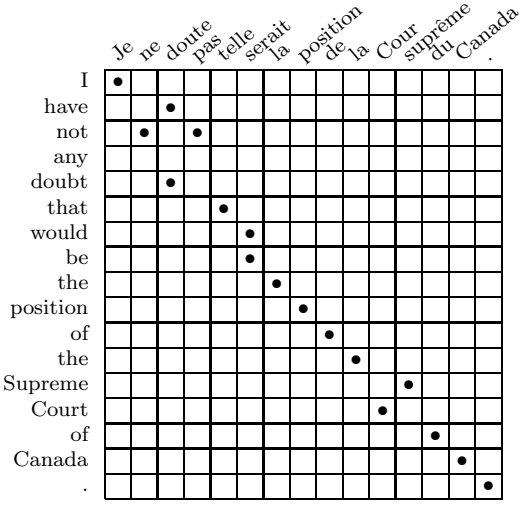


Figure 2: Visualization of word alignments with an alignment matrix.

presented in a single line or column. Pairs of long sentences therefore often cannot be shown entirely on the screen. Aligning pairs of long sentences then requires scrolling back and forth, especially when there are considerable differences in word order between the two languages. Moreover, as sentence length increases, visualization by drawing lines quickly becomes cluttered, and alignment matrices become hard to track. We believe that it is not only because of the intrinsic difficulties of explaining translations by word alignment but also because of such interface issues that aligning words manually has the reputation of being a very tedious task.

3 Yawat

Yawat (Yet Another Word Alignment Tool) was developed to remedy this situation by providing an efficient interface for creating and editing word alignments manually. It is implemented as web application with a thin CGI script on the server side and a browser-based[2] client written in JavaScript. This setup facilitates collaborative efforts with multiple annotators working remotely without the overhead of needing to organize the transfer of alignment data separately. The server-side data structure was deliberately kept small and simple, so that the tool or some of its components can be used as a visualization front-end for existing word alignments.

Yawat's most prominent distinguishing feature is

[2]Unfortunately, differences in the underlying DOM implementations make it laborious to implement truly browser-independent web applications in JavaScript. *Yawat* was developed for FireFox and currently won't work in Internet Explorer.

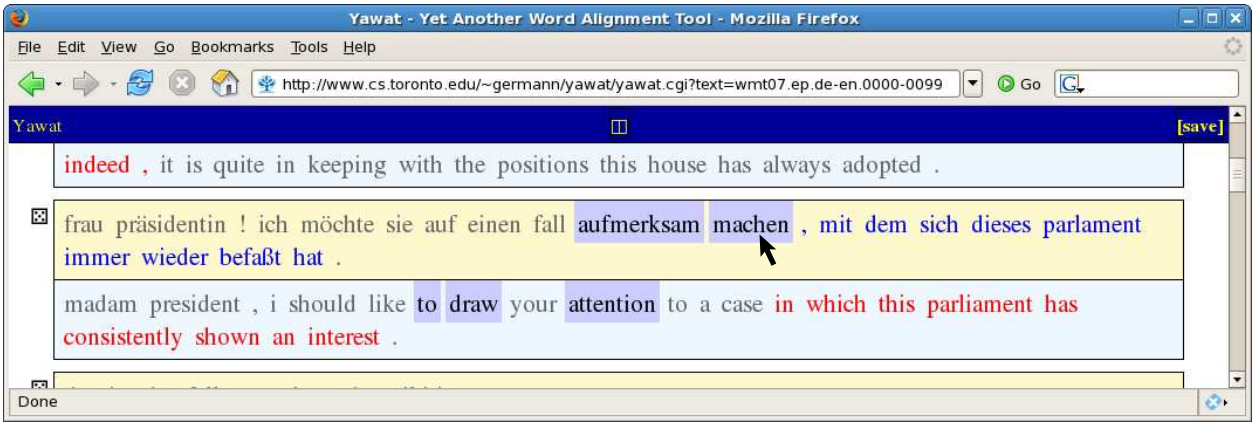


Figure 3: Alignment visualization with *Yawat*. As the mouse is moved over a word, the word and all words linked with it are highlighted. The highlighting is removed when the mouse leaves the word in question. This allows the annotator to focus on one item at a time, without any distracting visual clutter from other word alignments.

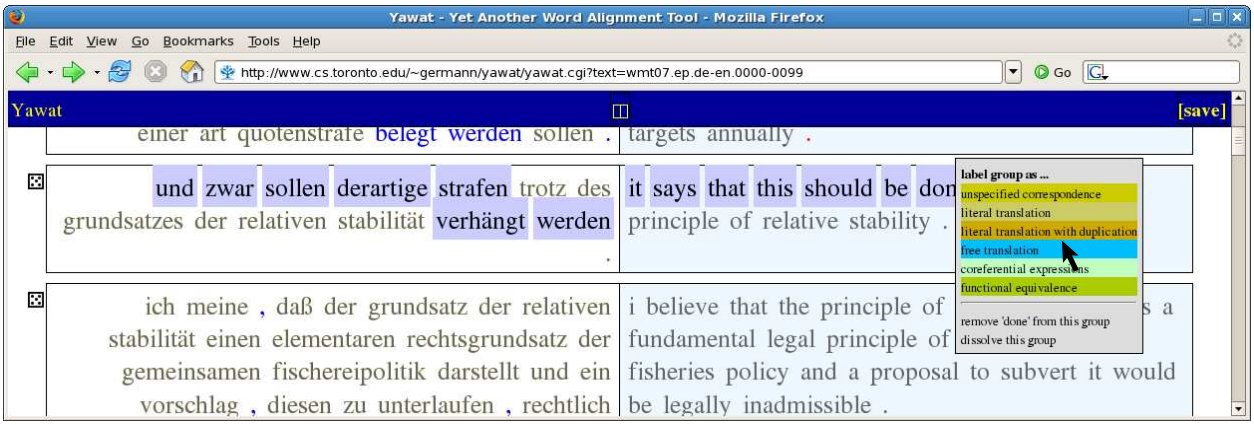


Figure 4: *Yawat* allows alignment relations to be labeled via context menues. Parallel text can be displayed side-by-side as in this screenshot or stacked as in Fig. 3.

the use of dynamic instead of static visualization. Rather than showing alignment links permanently by drawing lines or showing marks in an alignment matrix, associated words are shown only for one word at a time, as determined by the location of the mouse pointer. When the mouse is moved over a word in the text, the word and all the words associated with it are highlighted; when the mouse is moved away, the highlighting is removed. Figure 3 gives a snapshot of the tool in action.

Designed primarily as a tool for creating word alignments, one design objective was to minimize mouse travel required to align words. The interface therefore has no 'link words' button but uses mouse clicks on words directly to establish alignment links. A left-click on a word puts the tool into *edit* mode and opens an 'alignment group' (i.e., a set of words that supposedly constitute the expression of a concept in the two languages). Additional left-clicks on other words add them to or remove them from the current alignment group. A final right-click closes the group and puts the tool back into *view* mode. The typical case of aligning just two individual words thus takes only a single click on each of the two words: a left-click on the first word and a right-click on the second. As words are aligned, their color changes to indicate that they have been dealt with, so that the annotator can easily keep track of which words have been aligned, and which ones still need to be aligned. Notice the difference in color (or shading in a gray-scale printout) in the sentences in Fig. 3, whose first halves have been aligned while their latter halves are still unaligned.

In *view* mode, alignment groups can be labeled with a customizable set of tags via a context menu

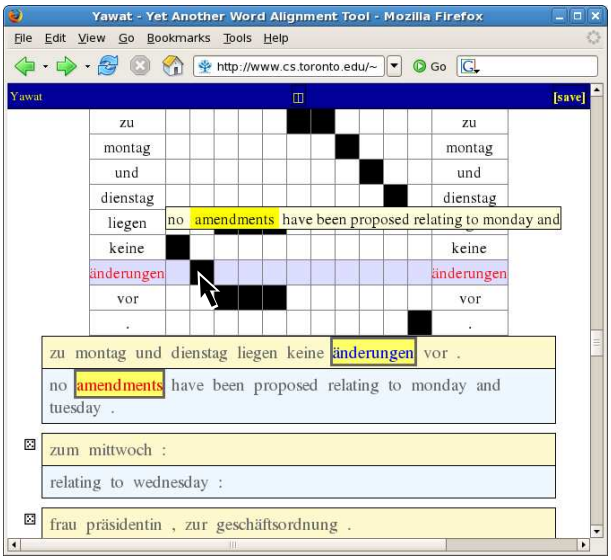


Figure 5: *Yawat* can also show alignments as alignment matrices. The tooltip-like floating bar above the mouse pointer provides column labels.

triggered by a right-click on a word (Fig. 4). For example, one might want to classify translational correspondences as 'literal', 'non-literal / free', or 'coreferential without intensional equivalence'. Different colors are used to indicate different types of alignment; color schemes and tag sets can be configured on the server side.

3.1 Alignment matrix display

One of the drawbacks of the dynamic visualization scheme employed in *Yawat* is that it provides no bird's-eye view of the overall alignment structure, as

it is provided by alignment matrices. We therefore decided to add alignment matrices as an additional visualization option. Alignment matrices are created on demand and can be switched on and off for each sentence pair. Word alignments can be edited in the alignment matrix view by clicking into the respective matrix cells to link or unlink words. Alignments matrices and the normal side-by-side or top-and-bottom display of the sentence pair in question are interlinked, so that an changes in the alignment matrix are immediately visible in the 'normal' display and vice versa (see Fig. 5).

4 Conclusion

We presented *Yawat*, a tool for the creation and visualization of word- and phrase alignments. An on-line demo is currently available at `http://www.cs.toronto.edu/~germann/yawat/yawat.cgi`. A package including the server-side scripts and the client-side code is available upon request.

References

Ahrenberg, Lars, Mikael Andersson, and Magnus Merkel. 2002. "A system for incremental and interactive word linking." *Third International Conference on Linguistic Resources and Evaluation (LREC-2002)*, 485–490. Las Palmas, Spain.

Daumé, Hal. "HandAlign." `http://www.cs.utah.edu/~hal/HandAlign/`.

Fraser, Alexander and Daniel Marcu. 2006. "Semi-supervised training for statistical word alignment." *Joint 44th Annual Meeting of the Association for Computational Linguistics and 21th International Conference on Computational Lignuistics (COLING-ACL '98)*, 769–776. Sydney, Australia.

Germann, Ulrich. 2007. "Two tools for creating and visualizing sub-sentential alignments of parallel text." *Linguistic Annotation Workshop (LAW '07)*, 121–124. Prague, Czech Republic.

Hwa, Rebecca and Nitin Madnani. 2004. "The umiacs word alignment interface." `http://www.umiacs.umd.edu/~nmadnani/alignment/forclip.htm`.

Lambert, Patrik. 2004. "Alignment set toolkit." `http://gps-tsc.upc.es/veu/personal/lambert/software/AlignmentSet.html`.

Melamed, I. Dan. 1998. *Manual Annotation of Translational Equivalence: The Blinker Project.* Technical Report 98-07, Institute for Research in Cognitive Science (IRCS), Philadelphia, PA.

Moore, Robert C., Wen-tau Yih, and Andreas Bode. 2006. "Improved discriminative bilingual word alignment." *Joint 44th Annual Meeting of the Association for Computational Linguistics and 21th International Conference on Computational Lignuistics (COLING-ACL '98)*, 513–520. Sydney, Australia.

Och, Franz Josef and Hermann Ney. 2003. "A systematic comparison of various statistical alignment models." *Computational Linguistics*, 29(1):19–51.

Rassier, Brian and Ted Pedersen. 2003. "Alpaco: Aligner for parallel corpora." `http://www.d.umn.edu/~tpederse/parallel.html`.

Smith, Noah A. and Michael E. Jahr. 2000. "Cairo: An alignment visualization tool." *Second International Conference on Linguistic Resources and Evaluation (LREC-2000).*

Taskar, Ben, Simon Lacoste-Julien, and Dan Klein. 2005. "A discriminative matching approach to word alignment." *Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP '05)*, 73–80. Morristown, NJ, USA.

Tiedemann, Jörg. "UPlug: Tools for linguistic corpus processing, word alignment and term extraction from parallel corpora." `http://stp.ling.uu.se/cgi-bin/joerg/Uplug`.

Tiedemann, Jörg. 2006. "ISA & ICA — Two web interfaces for interactive alignment of bitexts." *Fifth International Conference on Linguistic Resources and Evaluation (LREC-2006).* Genoa, Italy.

SIDE: The Summarization Integrated Development Environment

Moonyoung Kang, Sourish Chaudhuri, Mahesh Joshi, Carolyn P. Rosé

Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213 USA
moonyoun,schaudhu,maheshj,cprose@cs.cmu.edu

Abstract

In this type-II demo, we introduce SIDE[1] (the Summarization Integrated Development Environment), an infrastructure that facilitates construction of summaries tailored to the needs of the user. It aims to address the issue that there is no such thing as the perfect summary for all purposes. Rather, the quality of a summary is subjective, task dependent, and possibly specific to a user. The SIDE framework allows users flexibility in determining what they find more useful in a summary, both in terms of structure and content. As an educational tool, it has been successfully user tested by a class of 21 students in a graduate course on Summarization and Personal Information Management.

1 Introduction

A wide range of summarization systems have been developed in the past 40 years, beginning with early work in the Library sciences field. To this day, a great deal of research in summarization focuses on alternative methods for selecting subsets of text segments based on a variety of forms of rhetorical analysis and relevance rankings. Nevertheless, while there is much in common between approaches used for summarization in a variety of contexts, each new summarization project tends to include a new system development effort, because a general purpose, extensible framework for summarization has not been made available. As an example, Teufel and Moens' (2002) argue that the summarization strategy for scientific articles must be different from news articles because the former focus on novelty of information, are much longer and very different in structure.

A large proportion of summarization systems do not allow users to intervene in the summarization process so that the form of the summary could be tailored to the individual user's needs (Mieskes, M., Müller, C., & Strube, M., 2007). From the same document, many summaries can potentially be generated, and the most preferable one for one user will not, in general, be the same as what is preferred by a different user. The fact that users with similar backgrounds can have vastly differing information needs is highlighted by Paice and Jones' (1993) study where an informal sentence selection experiment had to be abandoned because the participants, who were agriculture experts, were too influenced by their research interests to agree with each other. However, summarization systems tend to appear as black boxes from the user's perspective and the users cannot specify what they would want in the summary.

SIDE is motivated by the two scenarios mentioned above - the absence of a common tool for generating summaries from different contexts, as well as the fact that different users might have different information needs from the same document. Bellotti (2005) discusses the problem of information overload in communication media such as e-mail and online discussion boards. The rapid growth of weblogs, wikis and dedicated information sources makes the problem of information overload more acute. It also means that summarization systems have the responsibility of taking into account the kind of information that its user would be interested in.

With SIDE, we attempt to give the user a greater say in deciding what kind of information and how much of it the user wants as part of his summary.

[1] The working system can be downloaded from http://www.cs.cmu.edu/~cprose/SIDE.html, and a video of an example of SIDE use can be found at http://ankara.lti.cs.cmu.edu/side/video.swf.

This project is supported by ONR Cognitive and Neural Sciences Division, Grant number N000140510043

Proceedings of the ACL-08: HLT Demo Session (Companion Volume), pages 24–27,
Columbus, June 2008.

In the following sections, we elaborate on the features of SIDE and its technical details.

2 Functionality

The design of SIDE is aimed at allowing the user as much involvement at every stage of the summary generation process as the user wishes. SIDE allows the user to select a set of documents to train the system upon, and to decide what aspects of input documents should be detected and used for making choices, particularly at the stage of selecting a subset of segments to preserve from the source documents. The other key feature of the development environment is that it allows developers to plug in custom modules using the Plugin Manager in the GUI. In this way, advanced users can extend the capabilities of SIDE for meeting their specific needs while still taking advantage of the existing, general purpose aspects of SIDE.

The subsequent sub-sections discuss individual parts of system behavior in greater detail at a conceptual level. Screen shots and more step by step discussion of how to use the GUI are given with the case study that outlines the demo script.

2.1 Filters

To train the system and create a model, the user has to define a filter. Defining a filter has 4 steps – creating annotated files with user-defined annotations, choosing feature sets to train (unigrams, bigrams etc), choosing evaluation metrics (Word Token Counter, TF-IDF) and choosing a classifier to train the system.

Annotating Files: The GUI allows the user to create a set of unstructured documents. The user can create folders and import sets of documents or individual documents. The GUI allows the user to view the documents in their original form; alternatively, the user can add it to the filter and segment it by sentence, paragraph, or by own definition. The user can define a set of annotations for each filter, and use those to annotate segments of the file. The system has sentence and paragraph segmenters built into it. The user can also define a segmenter and plug it in.

Feature Sets: The feature set panel allows the user to decide which features the user wants to use in training the model. It is built on top of TagHelper Tools (Donmez et al., 2005) and uses it to extract the features chosen by the user. The system has options for using unigrams, bigrams, Part-Of-Speech bigrams and punctuation built into it, and the user can specify whether they wish to apply stemming and/or stop word removal. Like the segmenters, if the user wants to use a specific feature to train, the user can plug in the feature extractor for the same through the GUI.

Evaluation Metrics: The evaluation metric decides how to order the sentences that are chosen to be part of the summary. In keeping with the plug-in architecture of the system, the user can define own metric and plug it into the system using the Plugin Manager.

Classifier: The user can decide which classifier to train the model with. This functionality is built on top of TagHelper Tools, which uses the Weka toolkit (Witten & Frank, 2005) to give users a set of classifiers to choose from. Once the system has been trained, the user can see the training results in a panel which provides a performance summary - including the kappa scores computed through 10-fold cross validation and the confusion matrix, the sets of features extracted from the text, and the settings that were used for training the model.

The user can choose the model for classifying segments in the target document. The user also can plug-in a machine learning algorithm to the system if necessary.

2.2 Summaries

Summaries are defined by Recipes that specify what types of segments should be included in the resulting summary, and how a subset of the ones that meet those requirements should be selected and then arranged. Earlier we discussed how filters are defined. One or more filters can be applied to a text so that each segment has one or more labels. These labels can then be used to index into a text. For example, a Recipe might specify using a logical expression such that only a subset of segments whose labels meet some specified set of constraints should be selected. The selected subset is then optionally ranked using a specified Evaluation metric. Finally, from this ranked list, some number or some percentage of segments will then finally be selected to be included in the resulting summary. The segments are then optionally re-ordered to the original document order before including them in the summary, which is then displayed to the user.

3 Case Study

The following subsections describe an example where the user starts with some unstructured documents and uses the system to generate a specification for a summary, which can then be applied to other similar documents.

We illustrate a script outline of our demo presentation. The demo shows how simple it is to move through the steps of configuring SIDE for a type of summary that a user would like to be able to generate. In order to demonstrate this, we will lead the user through an annotation task where we assign dialogue acts to turns in some tutoring dialogues. From this annotated data, we can generate summaries that pull out key actions of particular types. For example, perhaps we would like to look at all the instructions that the tutor has given to a student or all the questions the student has asked the tutor. The summarizing process consists of annotating training documents to define filters, deciding which features to use along with what machine learning algorithm to train the filters, training the actual filters, defining a summary in terms of the structured annotation that is accomplished by the defined filters, and finally, summarizing target files using the resulting configuration. The purpose of SIDE is to provide both an easy GUI interface for people who are not familiar with programming, and extensible, plug-and-play code for those who want to program and change SIDE into a more sophisticated and specialized type of summarizer. The demo will provide options for both novice users primarily interested in working with SIDE through its GUI interface and for more experienced users who would like to work with the code.

3.1 Using the GUI

The summarization process begins with loading unstructured training and testing documents. Next, filters are defined by adding training documents, segmenting each by choosing an automatic segmenter, and assigning annotations to the segments.

After a document is segmented, the segments are annotated with labels that classify segments using a user-defined coding scheme (Figure 1). Unannotated segments are later ignored during the training phase. Next, a set of feature types, such as unigrams, bigrams, part of speech bigrams, etc., are selected, which together will be used to build the feature space that will be input to a selected machine learning algorithms, or ensemble of algorithms. In this example, 'Punctuation' Feature Class Extractor, which can distinguish interrogative sentence, is selected and for 'Evaluation Metrics', 'Word Token Counter' is selected. Now, we train this model with an appropriate machine learning algorithm. In this example, J48 which is

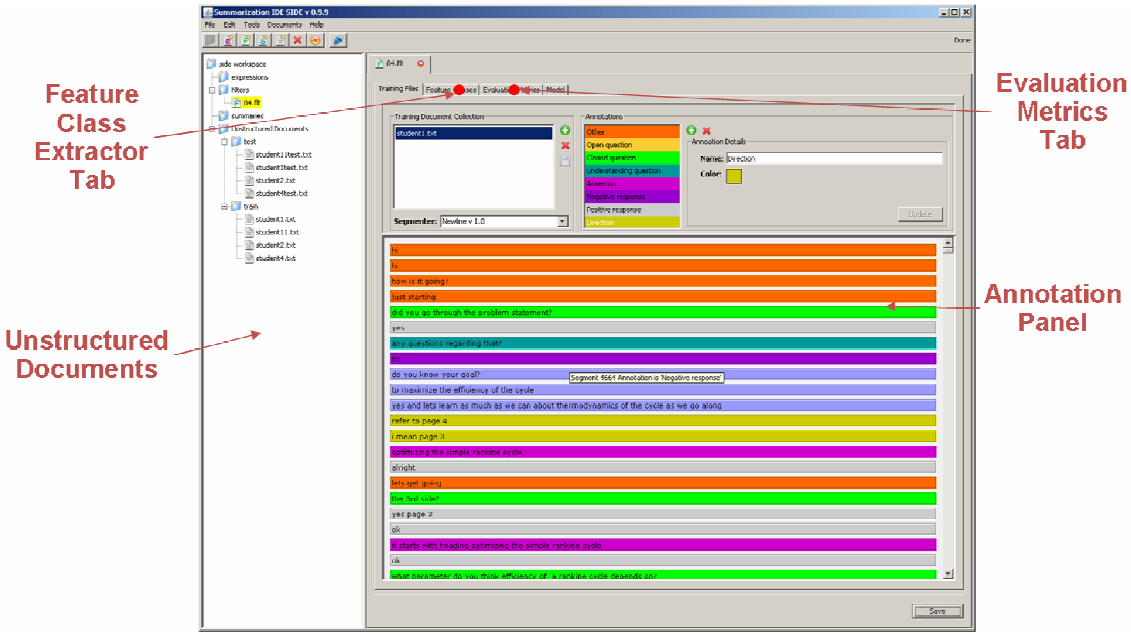


Figure 1: The interface where segments are annotated.

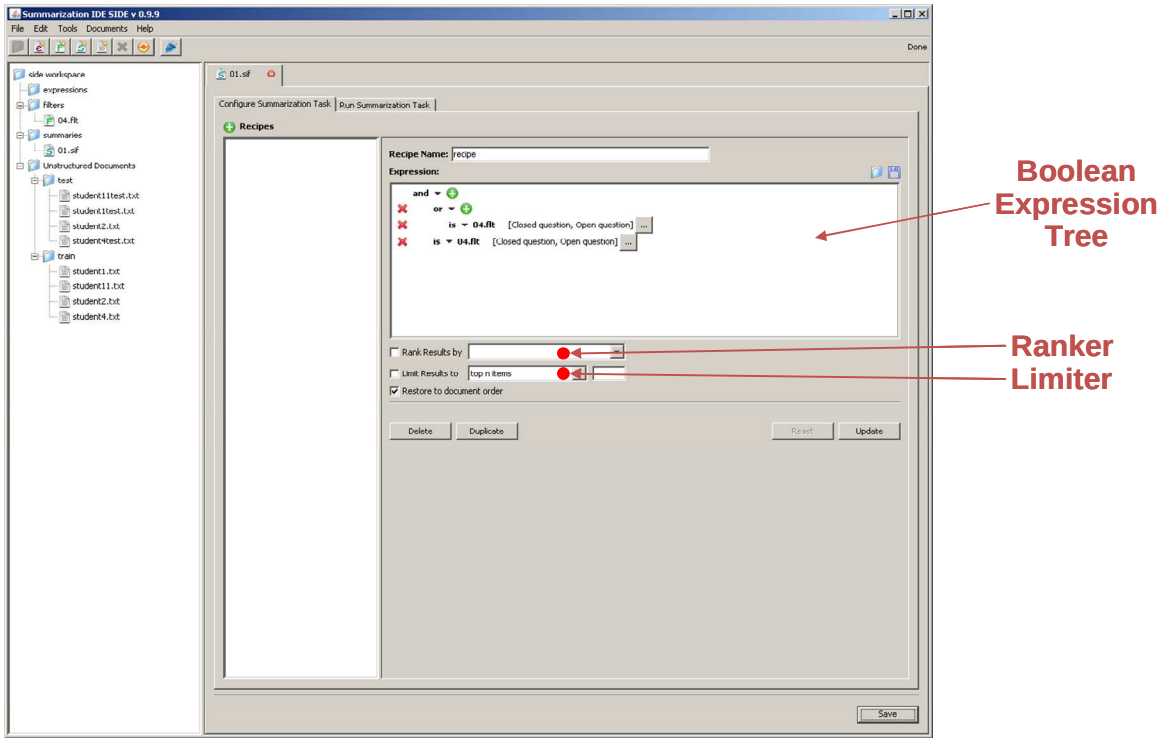


Figure 2: The interface for defining how to build a summary from the annotated data.

one of Weka's (Witten & Frank, 2005) decision tree learners is chosen as the learning algorithm. Users can explore different ensembles of machine learning algorithms, compare performance over the training data using cross-validation, and select the best performing one to use for summarization.

Once one or more filters have been defined, we must define how summaries are built from the structured representation that is built by the filters. Figure 2 shows the main interface for doing this. Recipes consist of four parts, namely 'Selecting', 'Ranking', 'Limiting', 'Sequencing'. Selection is done using a boolean expression tree consisting of 'and', 'or', and 'is' nodes. By doing selection, only those segments with proper annotations will be selected for inclusion in the resulting summary. Ranking is done by the Evaluation Metric selected when defining the Recipe. The size of a summary can be limited by limiting the number of segments you want in your summary. Finally, the summary can be reordered as you wish and displayed.

4 Current Directions

Currently, most of the functionality in SIDE focuses on the content selection problem. We acknowledge that to move beyond extractive forms of summarization, additional functionality at the summary generation stage is necessary. Our current work focuses on addressing these issues.

References

Bellotti, V., Ducheneaut, N., Howard, M., Smith, I., & Grinter, R. (2005). *Quality versus Quantity: E-Mail Centric Task Management and Its Relation with Overload*, Human-Computer Interaction, Volume 20,

Donmez, P., Rosé, C. P., Stegmann, K., Weinberger, A., and Fischer, F. (2005). *Supporting CSCL with Automatic Corpus Analysis Technology* , Proceedings of Computer Supported Collaborative Learning.

Mieskes, M., Müller, C., & Strube, M. (2007) *Improving extractive dialogue summarization by utilizing human feedback,* Proceedings of the 25th IASTED International Multi-Conference: artificial intelligence and applications, p.627-632

Paice, Chris D. & Jones, Paul A. (1993) *The identification of important concepts in highly structured technical papers. In* Proceedings of the 16th ACM-SIGIR Conference, pages 69–78

Teufel, S. & Moens, M. (2002). *Summarizing Scientific Articles: Experiments with Relevance and Rhetorical Status*, Computational Linguistics, Vol 28, No. 1.

Witten, Ian H.; Frank, Eibe (2005). *Data Mining: Practical machine learning tools and techniques, 2nd Edition*. Morgan Kaufmann, San Francisco.

ModelTalker Voice Recorder – An Interface System for Recording a Corpus of Speech for Synthesis

Debra Yarrington, John Gray, Chris Pennington

AgoraNet, Inc.
Newark, DE 19711
USA
{yarringt, gray, penningt}
@agora-net.com

H. Timothy Bunnell, Allegra Cornaglia, Jason Lilley, Kyoko Nagao, James Polikoff,

Speech Research Laboratory
A.I. DuPont Hospital for Children
Wilmington, DE 19803, USA
{bunnell, cornagli, lilley,
nagao, polikoff}@asel.udel.edu

Abstract

We will demonstrate the ModelTalker Voice Recorder (MT Voice Recorder) – an interface system that lets individuals record and bank a speech database for the creation of a synthetic voice. The system guides users through an automatic calibration process that sets pitch, amplitude, and silence. The system then prompts users with both visual (text-based) and auditory prompts. Each recording is screened for pitch, amplitude and pronunciation and users are given immediate feedback on the acceptability of each recording. Users can then rerecord an unacceptable utterance. Recordings are automatically labeled and saved and a speech database is created from these recordings. The system's intention is to make the process of recording a corpus of utterances relatively easy for those inexperienced in linguistic analysis. Ultimately, the recorded corpus and the resulting speech database is used for concatenative synthetic speech, thus allowing individuals at home or in clinics to create a synthetic voice in their own voice. The interface may prove useful for other purposes as well. The system facilitates the recording and labeling of large corpora of speech, making it useful for speech and linguistic research, and it provides immediate feedback on pronunciation, thus making it useful as a clinical learning tool.

1 Demonstration

1.1 MT Voice Recorder Background

While most of us are familiar with the highly intelligible but somewhat robotic sound of synthetic speech, for the approximately 2 million people in the United States with a limited ability to communicate vocally (Matas et al., 1985), these synthetic voices are inadequate. The restricted number of available voices lack the personalization they desire. While intelligibility is a priority for these individuals, almost equally important is the naturalness and individuality one associates with one's own voice. Individuals with difficulty speaking can be any age, gender, and from any part of the country, with regional dialects and idiosyncratic variations. Each individual deserves to speak with a voice that is not only intelligible, but uniquely his or her own. For those with degenerative diseases such as Amyotrophic Lateral Sclerosis (ALS), knowing they will be losing the voice that has become intricately associated with their identity is not only traumatic to the individual but to family and friends as well.

A form of synthesis that incorporates the qualities of individual voices is concatenative synthesis. In this type of synthesis, units of recorded speech are appended. By using recorded speech, many of the voice qualities of the person recording the speech remain in the resulting synthetic voice. Different synthesis systems append different sized

Proceedings of the ACL-08: HLT Demo Session (Companion Volume), pages 28–31,
Columbus, June 2008.

segments of speech. Appending larger the units of speech results in smoother, more natural sounding synthesis, but requires many hours of recording, often by a trained professional. The recording process is usually supervised, and the recordings are often hand-polished. Because appending smaller units requires less recording on the part of the speaker, this is the approach the ModelTalker Synthesizer has taken. However using smaller units may result in noticeable auditory glitches at concatenative junctures that are a result of variations (in pitch, amplitude, pronunciation, etc.) between the speech units being appended. Thus the speech recorded must be more uniform in pitch and amplitude. In addition, the units cannot be mispronounced because each unit is crucial to the resulting synthetic speech. In a smaller database there may not be a second example of a specific phoneme sequence.

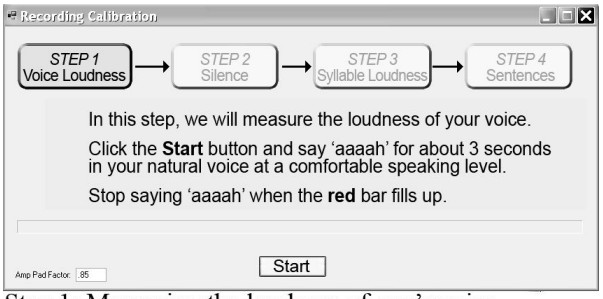
MT Voice Recorder expects that the individuals recording will be untrained and unsupervised, and may lack strength and endurance because of the presence of a degenerative disease. Thus the system is user-friendly enough for untrained, unsupervised individuals to record a corpus of speech. The system provides the user with feedback on the quality of each utterance they record in terms of pronunciation accuracy, relative uniformity of pitch, and relative uniformity of amplitude. Conference attendees will be able to experience this interface system and test all its different features.

1.2 Feature Demonstration

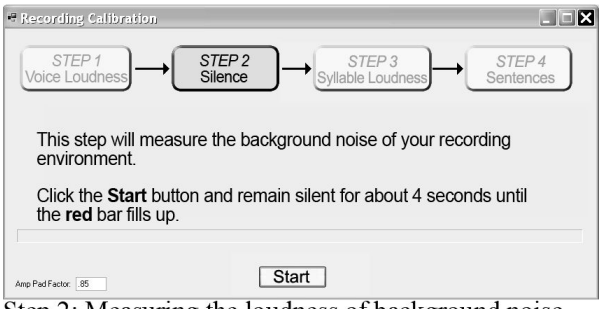
At the conference, attendees will be able to try out the different features of ModelTalker Voice Recorder. These features include automatic microphone calibration, pitch, amplitude, and pronunciation detection and feedback, and automatic phoneme labeling of speech recordings.

1.2.1 Microphone calibration

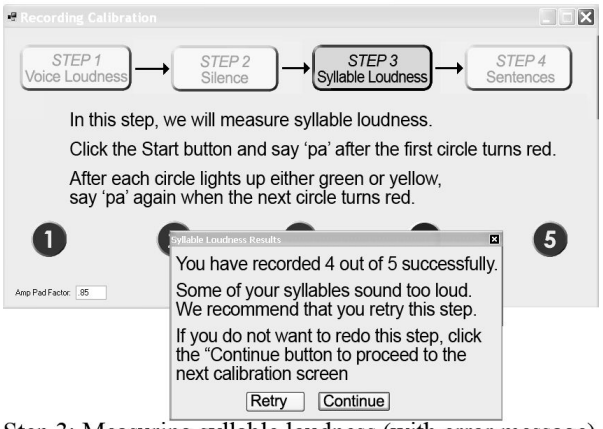
One important new feature of the MT Voice Recorder is the automatic microphone calibration procedure. In InvTool, a predecessor software of MT Voice Recorder, users had to set the microphone's amplitude. The system now calibrates the signal to noise ratio automatically through a step-by-step process (see Figure 1, below).



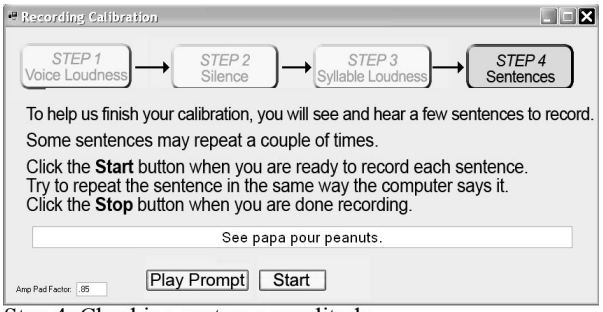
Step 1: Measuring the loudness of one's voice



Step 2: Measuring the loudness of background noise



Step 3: Measuring syllable loudness (with error message)



Step 4: Checking sentence amplitude

Figure 1: Automatic microphone calibration procedure

Using the automatic calibration procedure, the optimal signal to noise ratio is set for the recording session. These measurements are retained for future recording sessions in cases in which an indi-

vidual is unable to record the entire corpus in one sitting.

Once the user has completed the automatic calibration procedure, he will be able to start recording a corpus of speech. The interface has been designed with the assumption that individuals will be recording without supervision. Thus the interface incorporates a number of feedback mechanisms to aid individuals in making a high quality corpus for synthesis (see Figure 2, below).

1.2.2 Recording Utterances

The corpus was carefully chosen so that all frequently used phoneme combinations are included at least once. Thus it is critical that users pronounce prompted sentences in the manner in which the system expects. Alterations in pronunciation as small as saying /i/ versus /ə/ for "the," for example, can negatively affect the resulting synthetic voice. To reduce the incidence of alternate pronunciation, the user is prompted with both a text and an auditory version of the utterance.

1.2.3 Recording Feedback

Once an utterance has been recorded, the user receives feedback on the overall quality of the utterance. Specifically, the user receives feedback on the pitch, the overall amplitude, and the pronunciation of the recording.

Pitch: The user receives feedback on whether the utterance's average pitch is within range of the user's base pitch determined during the calibration process. Collecting all recordings within a relatively small pitch range minimizes concatenation costs during the synthesis process. MT Voice Recorder determines the average pitch of each utterance and gives the user feedback on whether the pitch is within an acceptable range. This feedback mechanism also helps to eliminate cases in which the system is unable to accurately track the pitch of an utterance. In these cases, the utterance will be marked unacceptable and the user should rerecord, hopefully yielding an utterance with more accurate pitch tracking.

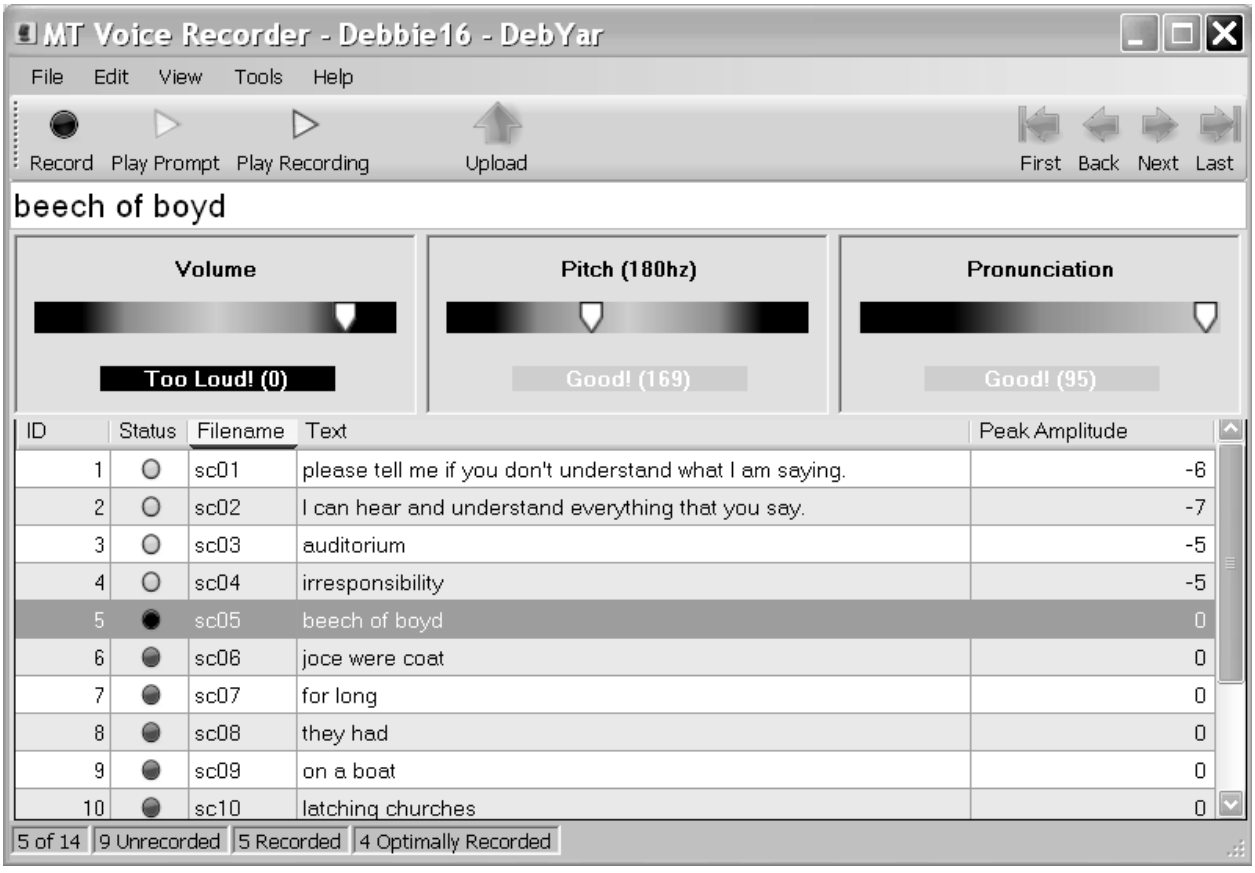


Figure 2: MT Voice Recorder User Interface

Amplitude: The user is also given feedback on the overall amplitude of an utterance. If the amplitude is either too low or too high, the user must rerecord the utterance.

Pronunciation: Each recorded utterance is evaluated for pronunciation. Each utterance within the corpus is associated with a string of phonemes representing its transcription. When an utterance is recorded, the phoneme string associated with the utterance is force-aligned with the recorded speech. If the alignment does not fall within an acceptable range, the user is given feedback that the recording's pronunciation may not be acceptable and the user is given the option of rerecording the utterance.

1.2.4 Automatic Phoneme Labeling

During the process of pronunciation evaluation, an associated phoneme transcription is aligned with the utterance. This alignment is retained so that each utterance is automatically labeled. Once the entire corpus has been recorded, alignments are automatically refined based on specific individual voice characteristics.

1.2.5 Other Features

The MT Voice Recorder also allows users to add utterances of their choice to the corpus of speech for the synthetic voice. These utterances are those the user wants to be synthesized clearly and will automatically be included in their entirety in the speech database. These utterances are also automatically labeled before being stored.

In addition, for those with more speech and linguistic experience, the system has a number of other features that can be explored. For example, the MT Voice Recorder also allows one to change settings so that the phoneme string, peak amplitude, RMS range, average F0, F0 range, and pronunciation score can be viewed. Users may use this information to more precisely adjust their utterances.

1.3 Synthetic Voice Demonstration

Those attending the demonstration will also be able to listen to a sampling of synthetic voices created using the ModelTalker system. While one of the synthetic voices was created by a professional speaker and manually polished, all other voices were created by untrained individuals, most of whom have ALS, in an untrained setting, with the recordings having no manual polishing.

2 Other Applications

Although the MTVR was designed specifically to record speech for the creation of a database that will be used in speech synthesis, it can also be used as a digital audio recording tool for speech research. For example, the MT Voice Recorder offers useful features for language documentation. An immediate warning about a poor quality recording will alert a researcher to rerecord the utterance. MT Voice Recorder employs file formats that are recommended for digital language documentation (e.g., XML, WAV, and TXT) (Bird & Simons, 2003). The recorded files are automatically stored with broad phonetic labels. The automatic saving function will reduce the time of recordings and the potential risk for miscataloging the files. Currently, the automatic phonetic labeling feature is only available for English, but it could be applicable to different languages in the future.

For more information about the ModelTalker System and to experience an interactive demo as well as listen to sample synthetic voices, visit http://www.modeltalker.com.

Acknowledgments

This work was supported by STTR grants R41/R42-DC006193 from NIH/NIDCD and from Nemours Biomedical Research. We are especially indebted to the many people with ALS, the AAC specialists in clinics, and other interested individuals who have invested a great deal of time and effort into this project and have provided valuable feedback.

References

Bird, S. and Simons, G.F. (2003). Seven dimensions of portability for language documentation and description. *Language*, *79*(3): 557-582.

Matas, J., Mathy-Laikko, P., Beaukelman, D. and Legresley. K. (1985). Identifying the nonspeaking population: a demographic study, *Augmentative & Alternative Communication*, *1*: 17-31.

The QuALiM Question Answering Demo: Supplementing Answers with Paragraphs drawn from Wikipedia

Michael Kaisser
School of Informatics
University of Edinburgh
M.Kaisser@sms.ed.ac.uk

Abstract

This paper describes the online demo of the QuALiM Question Answering system. While the system actually gets answers from the web by querying major search engines, during presentation answers are supplemented with relevant passages from Wikipedia. We believe that this additional information improves a user's search experience.

1 Introduction

This paper describes the online demo of the QuALiM[1] Question Answering system (http://demos.inf.ed.ac.uk:8080/qualim/). We will refrain from describing QuALiM's answer finding strategies–our work on QuALiM has been described in several papers in the last few years, especially Kaisser and Becker (2004) and Kaisser et al. (2006) are suitable to get an overview over the system–but concentrate on one new feature that was developed especially for this web demo: In order to improve user benefit, answers are supplemented with relevant passages from the online encyclopedia Wikipedia. We see two main benefits:

1. Users are presented with additional information closely related to their actual information need and thus of potential high interest.

2. The returned text passages present the answer in context and thus help users to validate the answer–there always will be the odd case where a system returns a wrong result.

[1] for ***Qu**estion **A**nswering with **Li**nguistic **M**ethods*

Historically, our system is web-based, receiving its answers by querying major search engines and post processing their results. In order to satisfy TREC requirements–which require participants to return the ID of one document from the AQUAINT corpus that supports the answer itself (Voorhees, 2004)–we already experimented with answer projection strategies in our TREC participations in recent years. For this web demo we use Wikipedia instead of the AQUAINT corpus for several reasons:

1. QuALiM is an open domain Question Answering system and Wikipedia is an "open domain" Encyclopedia; it aims to cover *all* areas of interest as long as they are of *some* general interest.

2. Wikipedia is a free online encyclopedia. Other than the AQUAINT corpus, there are no legal problems when using it for a public demo.

3. Wikipedia is frequently updated, whereas the AQUAINT corpus remains static and thus contains a lot of outdated information.

Another advantage of Wikipedia is that the information contained is much more structured. As we will see, this structure can be exploited to improve performance when finding answers or–as in our case–projecting answers.

2 How Best to Present Answers?

In the fields of Question Answering and Web Search, the issue how answers/results should be presented is a vital one. Nevertheless, as of today, the majority of QA system–which a few notable exceptions, e.g. MIT's START (Katz et al., 2002)–are

Proceedings of the ACL-08: HLT Demo Session (Companion Volume), pages 32–35,
Columbus, June 2008.

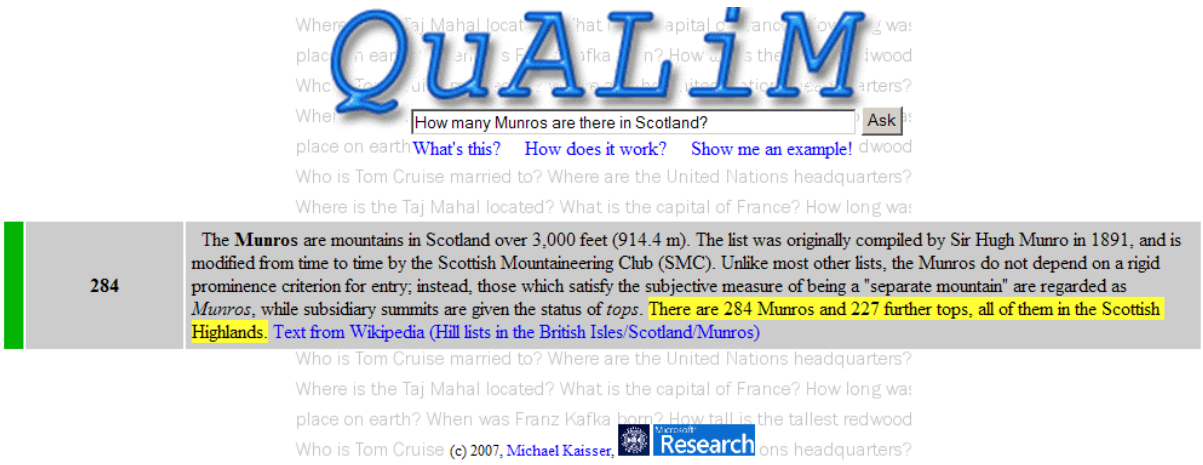


Figure 1: Screenshot of QuALiM's response to the question "How many Munros are there in Scotland?" The green bar to the left indicates that the system is confident to have found the right answer, which is shown in bold: "284". Furthermore, one Wikipedia paragraph which contains additional information of potential interest to the user is displayed. In this paragraph the sentence containing the answer is highlighted. This display of context also allows the user to validate the answer.

still experimental and research-oriented and typically only return the answer itself. Yet it is highly doubtful that this is the best strategy.

Lin et al. (2003) performed a study with 32 computer science students comparing four types of answer context: exact answer, answer-in-sentence, answer-in-paragraph, and answer-in-document. Since they were interested in interface design, they worked with a system that answered all questions correctly. They found that 53% of all participants preferred paragraph-sized chunks, 23% preferred full documents, 20% preferred sentences, and one participant preferred exact answer.

Web search engines typically show results as a list of titles and short snippets that summarize how the retrieved document is related to the query terms, often called *query-biased summaries* (Tombros and Sanderson, 1998). Recently, Kaisser et al. (2008) conducted a study to test whether users would prefer search engine results of different lengths (phrase, sentence, paragraph, section or article) and whether the optimal response length could be predicted by human judges. They find that judges indeed prefer different response lengths for different types of queries and that these can be predicted by other judges.

In this demo, we opted for a slightly different, yet related approach: The system does not decide on one answer length, but always presents a combination of three different lengths to the user (see Figure 1): The answer itself (usually a *phrase*), is presented in bold. Additionally, a *paragraph* relating the answer to the question is shown, and in this paragraph one *sentence* containing the answer is highlighted. Note also, that each paragraph contains a link that takes the user to the Wikipedia *article*, should he/she want to know more about the subject. The intention behind this mode of presentation is to prominently display the piece of information the user is most interested in, but also to present context information and to furthermore provide options for the user to find out more about the topic, should he/she want to.

3 Finding Supportive Wikipedia Paragraphs

We use Lucene (Hatcher and Gospodnetić, 2004) to index the publically available Wikipedia dumps (see http://download.wikimedia.org/). The text inside the dump is broken down into paragraphs and each paragraph functions as a Lucene *document*. The data of each paragraph is stored in three fields: *Title*, which contains the title of the Wikipedia article the paragraph is from, *Headers*, which lists the title and all section and subsection headings indicating the position of the paragraph in the article and *Text*, which stores the text of the article. An example can be seen

in Table 1.

Title	"Tom Cruise"
Headers	"Tom Cruise/Relationships and personal life/Katie Holmes"
Text	"In April 2005, Cruise began dating Katie Holmes ... the couple married in Bracciano, Italy on November 18, 2006."

Table 1: Example of Lucene index fields used.

As mentioned, QuALiM finds answers by querying major search engines. After post processing, a list of answer candidates, each one associated with a confidence value, is output. For the question "Who is Tom Cruise married to?", for example, we get:

```
81.0: "Katie Holmes"
35.0: "Nicole Kidman"
```

The way we find supporting paragraphs for these answers is probably best explained by giving an example. Figure 3 shows the Lucene query we use for the mentioned question and answer candidates. (The numbers behind the terms indicate query weights.) As can be seen, we initially build two separate queries for the *Headers* and the *Text* fields (compare Table 1). In a later processing step, both queries are combined into a single query using Lucene's `MultipleFieldQueryCreator` class. Note also that both answer candidates ("Katie Holmes" and "Nicole Kidman") are included in this one query. This is done because of speed issues: In our setup, each query takes up roughly two seconds of processing time. The complexity and length of a query on the other hand has very little impact on speed.

The type of question influences the query building process in a fundamental manner. For the question "When was Franz Kafka born?" and the correct answer "July 3, 1883", for example, it is reasonable to search for an article with title "Franz Kafka" and to expect the answer in the text on that page. For the question "Who invented the automobile?" on the other hand, it is more reasonable to search the information on a page called "Karl Benz" (the answer to the question). In order to capture this behaviour we developed a set of rules that for different type of questions, increases or decreases constituents' weights in either the *Headers* or the *Text* field.

Additionally, during question analysis, certain question constituents are marked as either *Topic* or *Focus* (see Moldovan et al., (1999)). For the earlier example question "Tom Cruise" becomes the *Topic* while "married" is marked *Focus*[2]. These also influence constituents' weights in the different fields:

- Constituents marked as *Topic* are generally expected to be found in the *Headers* field. After all, the topic marks *what the question is about*. In a similar manner, titles and subtitles help to structure an article, assisting the user to navigate to the place where the relevant information is most likely to be found: A paragraph's titles and subtitles indicate *what the paragraph is about*.
- Constituents marked as *Focus* are generally expected to be found in the text, especially if they are verbs. The focus indicates what the question asks for, and such information can usually rather be expected in the text than in titles or subtitles.

Figure 3 also shows that, if we recognize named entities (especially person names) in the question or answer strings, we once include each named entity as a quoted string and additionally add the words it contains separately. This is to boost documents which contain the complete name as used in the question or the answer, but also to allow documents which contain variants of these names, e.g. "Thomas Cruise Mapother IV".

The formula to determine the exact boost factor for each query term is complex and a matter of ongoing development. It additionally depends on the following criteria:

- Named entities receive a higher weight.
- Capitalized words or constituents receive a higher weight.
- The confidence value associated with the answer candidate influences the boost factor.
- Whether a term originates from the question or an answer candidate influences its weight in a different manner for the header and text fields.

[2]With allowing verbs to be the *Focus*, we slightly depart from the traditional definition of the term.

Header query:

```
"Tom Cruise"^10 Tom^5 Cruise^5 "Katie Holmes"^5 Katie^2.5 Holmes2.^5
"Nicole Kidman"^4.3 Nicole^2.2 Kidman^2.2
```

Text query:

```
married^10 "Tom Cruise"^1.5 Tom^4.5 Cruise^4.5 "Katie Holmes"^3 Katie^9 Holmes^9
"Nicole Kidman"^2.2 Nicole^6.6 Kidman^6.6
```

Figure 2: Lucene Queries used to find supporting documents for the "Who is Tom Cruise married to?" and the two answers "Katie Holmes" and "Nicole Kidman". Both queries are combined using Lucene's MultipleFieldQueryCreator class.

4 Future Work

Although QuALiM performed well in recent TREC evaluations, improving precision and recall will of course always be on our agenda. Beside this we currently focus on increasing processing speed. At the time of writing, the web demo runs on a server with a single 3GHz Intel Pentium D dual core processor and 2Gb SDRAM. At times, the machine is shared with other demos and applications. This makes reliable figures about speed difficult to produce, but from our log files we can see that users usually wait between three and twelve seconds for the system's results. While this is okay for a research demo, it definitely would not be fast enough for a commercial product. Three factors contribute with roughly equal weight to the speed issue:

1. Search engine's APIs usually do not return results as fast as their web interfaces built for human use do. Google for example has a built-in one second delay for each query asked. The demo usually sends out between one and four queries per question, thus getting results from Google alone takes between one and four seconds.

2. All received results need to be post-processed, the most computing heavy step here is parsing.

3. Finally, the local (8.3 GB big) Wikipedia index needs to be queried, which roughly takes two seconds per query.

We are currently looking into possibilities to improve all of the above issues.

Acknowledgements

This work was supported by Microsoft Research through the European PhD Scholarship Programme.

References

Erik Hatcher and Otis Gospodnetić. 2004. *Lucene in Action*. Manning Publications Co.

Michael Kaisser and Tilman Becker. 2004. Question Answering by Searching Large Corpora with Linguistic Methods. In *The Proceedings of the 2004 Edition of the Text REtrieval Conference, TREC 2004*.

Michael Kaisser, Silke Scheible, and Bonnie Webber. 2006. Experiments at the University of Edinburgh for the TREC 2006 QA track. In *The Proceedings of the 2006 Edition of the Text REtrieval Conference, TREC 2006*.

Michael Kaisser, Marti Hearst, and John Lowe. 2008. Improving Search Result Quality by Customizing Summary Lengths. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics*.

Boris Katz, Jimmy Lin, and Sue Felshin. 2002. The START multimedia information system: Current technology and future directions. In *Proceedings of the International Workshop on Multimedia Information Systems (MIS 2002)*.

Jimmy Lin, Dennis Quan, Vineet Sinha, Karun Bakshi, David Huynh, Boris Katz, and David R. Karger. 2003. What Makes a Good Answer? The Role of Context in Question Answering. *Human-Computer Interaction (INTERACT 2003)*.

Dan Moldovan, Sanda Harabagiu, Marius Pasca, Rada Mihalcea, Richard Goodrum, Roxana Girju, and Vasile Rus. 1999. LASSO: A tool for surfing the answer net. In *Proceedings of the Eighth Text Retrieval Conference (TREC-8)*.

A. Tombros and M. Sanderson. 1998. Advantages of query biased summaries in information retrieval. *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 2–10.

Ellen M. Voorhees. 2004. Overview of the TREC 2003 Question Answering Track. In *The Proceedings of the 2003 Edition of the Text REtrieval Conference, TREC 2003*.

Author Index