Programme

01/06/2008	**Workshop on Natural Language Processing resources, algorithms and tools for authoring aids**
9:00	***Welcome and Introduction***
9:30	**Invited Talk:** *Authoring tools developed at Microsoft: that was then, this is now* Takako Aikawa (Microsoft Research)
10h30	Morning coffee break
	Proposal for a Foreign Language Drafting Aid Elliott Macklovitch, Guy Lapalme
	Language-aware Text Editing Cerstin Mahlow, Michael Piotrowski, Michael Hess
	Enhancing the OpenOffice.org Word Processor with Natural Language Processing Capabilities Thomas Gitzinger, René Witte
	A Personalized Recommender System for Writing in the Internet Age Mari Carmen Puerta Melguizo, Olga Muñoz Ramos, Lou Boves, Toine Bogers, Antal van den Bosch
	Towards Automatic Document Quality Assurance Neil Newbold, Lee Gillam
13h05	***Panel session: Possible future directions for authoring aids***
13h30	End of workshop

Organisers

Robert Dale	Macquarie University, Sydney, Australia
Aurélien Max	LIMSI-CNRS, Université Paris-Sud 11, Orsay, France
Michael Zock	LIF, Marseille, France

Programme Committee

John Bateman	Universitat Bremen, Germany
Nadjet Bouayad-Agha	Pompeu Fabra University, Barcelona, Spain
Eric Brunelle	Druide Informatique, Montréal, Canada
Dan Cristea	University of Iasi, Romania
Robert Dale	Macquarie University, Sydney, Australia
Marc Dymetman	XRCE, Grenoble, France
Patrice Enjalbert	GREYC, Université de Caen, France
Nicolas Hernandez	LINA, Université de Nantes, France
Graeme Hirst	University of Toronto, Canada
Eduard Hovy	ISI, University of Southern California, USA
Aurélien Max	LIMSI, Université Paris-Sud 11, Orsay, France
Thierry Olive	LMDC, Poitiers, France
Marie-Paule Péry-Woodley	ERSS, Université de Toulouse 2, France
Gisela Redeker	University of Groningen, The Netherlands
Ehud Reiter	University of Aberdeen, UK
Michael Zock	LIF, Marseille, France

Table of Contents

Oral presentations

Author Index

Foreword

The research results contained in the present volume address a number of important issues around the technicalities of authoring processes and how and under which conditions authoring aids might be of help to human writers.

Nowadays, writers and their authoring tools manipulate text at different levels: whereas writers have ideas and linguistic means to express them, word processors only handle characters and strings. This mismatch can be a source of frustration on the part of the writer, who might have to repeat apparently simple tasks such as pluralizing a noun phrase and adapting the governed verb accordingly, without any means to have the system do it at some later point. Cerstin Mahlow, Michael Piotrowski and Michael Hess describe language-aware editing functions and their practical implementation within a text editor, and make steps towards a taxonomy of revising and editing operations. This type of work poses interesting questions regarding how and when such assistance could be used, and for which types of linguistic units.

Starting from the same hypothesis that integrating language analysis functions can be useful to writers if used appropriately, Thomas Gitzinger and René Witte propose a service-oriented approach based on open standards and open source tools whereby NLP services can be integrated as plug-ins into word processors, and thus be called directly by writers. Interesting perspectives stemming from this work include inferring automatically when automatic analysis could be done by the system, for example based on the user's behavior, even when not executed explicitly by the user. Such behavior from future authoring aids will certainly have a positive impact on the work of writers, but it is of course crucial to ensure that the user can understand what is being done by the system and accept it.

If writers do not want to receive help when they believe it is not appropriate, some help might be particularly welcome if it corresponds to a real need and comes at the right moment. The paper by Mari Carmen Puerta Melguizo, Olga Muñoz Ramos, Lou Boves, Toine Bogers and Antal van den Bosch describes the integration of a proactive recommender system into a word processor to assist writers in searching for relevant documents from the Internet while writing text. They show that using profiles to ensure that the presented information is relevant to the writer and to the text being written can lead to improvements in text quality.

The way in which documents are written has a direct impact on how readers will be able to read and understand them. Neil Newbold and Lee Gillam discuss how techniques from corpus linguistics can be used by writers during document production and revision for document quality assurance, for example in terms of language simplicity and consistency. They discuss the limits of readability formulae and consider terminological complexity and its use to compute the cognitive load in understanding a new text.

Lastly, Guy Lapalme and Elliott Maklovitch consider cases in which a computer can assist a writer in actually writing text. They first review natural contexts of use when an underlying meaning representation or a source text in another language exists, and present an interactive tool for translators. Then, they propose a similar interactive-predictive approach for a drafting assistant which could prove useful for people writing in a foreign language. Such a tool could help in writing better sentences conforming to language models and in proposing multi-word completions. Again, the fact that such a system could learn from its interaction with the user is discussed as an essential element for its acceptance by its intended users.

When we drafted the call for papers, we sought papers on all aspects of the text generation process. There were a number of areas which we consider important in thinking about the writing process which were not represented in the submissions: specifically, more conceptually-oriented aspects of the process such as content determination, outline planning, and lexical access did not feature. We suspect there is little work in these areas within the community that normally attends LREC, but there are areas outside of natural language processing where these topics are under debate. Clearly a way to move this part of the field forward is to reach out to these other communities, and in particular to finding ways of drawing in both professional writers (such as journalists) and psychologists. We look forward to the challenge of growing the debate into a truly interdisciplinary enterprise.

Robert Dale, Aurélien Max, Michael Zock

Oral presentations

Proposal for a Foreign Language Drafting Aid

Guy Lapalme, Elliott Macklovitch
Laboratoire RALI, Université de Montréal
Montréal, Canada
E-mail: lapalme@iro.umontreal.ca, macklovi@iro.umontreal.ca

Abstract

We examine a range of existing authoring aids (understood in the widest sense), from full-scale NLG to a new type of interactive MT. All of these depend on an independently available meaning representation, either a language-neutral formal model or a parallel text in another language. When no such meaning representation is available, however, current authoring aids can offer little more than word completion, as in augmentative communication systems or SMS. We consider the particular situation of a domain specialist who is called upon to draft a text in a second or a foreign language, and propose ways in which a new authoring tool might be able to assist him/her by suggesting multi-word completions, which should be more beneficial.

1. Introduction

The natural language generation (NLG) process is often separated into two phases: *What to say?* which corresponds to the building of a semantic representation, and *How to say it?* which chooses the words and sentences to convey the meaning encoded in that semantic representation. Although these phases have been defined primarily for engineering the modules of automatic text generators, they also have cognitive motivation and, informally at least, seem to correspond to the way humans plan and produce their communications.

Automated aids designed for human authors can also benefit from this division of the writing task. In certain situations where the intended meaning of a projected text is independently available, automated aids can be developed that provide the author with considerable assistance in generating the desired text. We will consider two such contexts in some detail. The first is where a non-linguistic, fully explicit, formal meaning representation is independently available. In this case, current NLG techniques allow for the automatic generation of certain types of texts directly from the data, and even in multiple languages. The other context is the case of translation, where the source text – a little like in painting by numbers – provides the semantic outline of the desired target text and the tones of the message to be transmitted. The problem with using a natural language text as a meaning representation is that it is rarely wholly unambiguous: not only do most source texts generally under-determine the meaning required to achieve a high-quality translation, but they often allow for multiple, equally acceptable translations. This remains the principal obstacle to fully automatic, high-quality machine translation (MT) today, and, despite impressive recent progress in this field, it is not likely to change for some time. Hence, where publication quality is a sine qua non, the only viable alternative is to retain the services of a human, either as a post-editor of the raw MT output, or as a translator who will hopefully have access to automated support tools. The former approach (post-editing) is not always cost-effective and humans often dislike it; the latter is generally referred to as machine-aided human translation (MAHT). In this paper, we will report on a novel form of MAHT that proposes multi-word completions of the target text, which the translator may accept or reject in whole or in part.

However, where *no* meaning representation (linguistic or otherwise) is independently available – and this corresponds to the vast majority of text drafting situations today – current authoring aids can offer little more than simple word completion. The central question we want to raise in this paper is whether it would be possible to improve on this, and if so how.

2. Text Generation

Canadian weather bulletins provide an interesting example of the case where a fully specified, non-linguistic knowledge base is available, and this for a number of reasons. First, the text of these simple bulletins, which is derived from the numerical content of huge meteorological databases, is highly stereotypical in form. For many years, these short texts were manually drafted by expert meteorologists and then translated into English or French by human translators. Beginning in 1977, the MÉTÉO system (Chandioux & Guéraud, 1981) succeeded in largely removing the human translators from the loop,[1] thereby providing MT with one of its first major success stories; but the meteorologists were still required to draft the source texts that were automatically translated by MÉTÉO. More recently, however, NLG techniques have been developed that allow for the parallel generation, directly from the numerical databases, of a large subset of these weather bulletins in both English and French, thereby dispensing with the need for the manual drafting of the text in one language and its machine translation into the other. Originally developed over a decade ago (c.f. Goldberg et al., 1994), this technology has only recently come into full daily use at Environment Canada, and still not for all types of public weather forecasts.

It turns out, however, that the particular constellation of factors that make weather bulletins such an ideal application for both MT and NLG – the rigid and highly restricted sublanguage in which they are drafted; the large volume and steady rate at which they are published; and

[1] Some translators were still required to revise the system's output.

the independent existence of a formal database from which they are derived – is relatively rare, or at least in this combination. In most other cases where an underlying knowledge base exists independently of the text generation process, the type of texts required are rarely as entirely predictable as weather bulletins are. Hence, even in such sophisticated NLG systems as MDA (Lapalme et al., 2003), the interactive composition of a text describing a pharmaceutical product involves a fair amount of word (or concept) selection from menus that list permissible completions or extensions, according to a grammar and a pre-established semantic hierarchy. However, once the author's intentions have been fully specified, ideally in a language-independent manner, the system may then be able to generate parallel versions of the desired text in multiple languages. But the situation here is quite different from that of weather bulletins, if only because there isn't a constantly evolving demand for this kind of pharmaceutical documentation.[2]

Hartley & Paris (1997) argue that in certain high-tech industries where large volumes of multilingual documentation are required, a trend has begun to emerge which eschews machine translation in favour of multilingual NLG. They also point to a concomitant blurring of the distinction between a translator and a technical writer, whereby a major part of what they call the *authoring* task now involves "the specification of the conceptual model (or message) underlying task-oriented instructions from which the documentation itself can be written." (p.115) Hartley & Paris convincingly demonstrate the advantages that this approach offers over sequential translation from a source text, although they admit that these advantages crucially hinge on the availability of an underlying knowledge base. "If such a knowledge base already exists or can be constructed automatically, then multilingual generation is a viable option for producing multilingual documents." (p.117) And indeed, there has been much exciting work in NLG over the last decade that is devoted to precisely this problem, i.e. finding ways to assist domain experts who do not have the skills or the time to master formal knowledge representation systems; see in particular the various implementations of the *what you see is what you mean* approach, e.g. (Power & Scott, 1998). To call this kind of work *authoring*, however, seems somewhat abusive, or at least at variance with the most widespread definition of this verb, which (according to Encarta's online dictionary) means, "to write or be responsible for the final form of a book, report, or other **text**" (our emphasis). Be that as it may, in the great majority of text drafting situations, a fully specified knowledge base is not available and the construction of one is simply not a viable option; hence, full-scale NLG is just not applicable in these situations.

3. Predictive Text Technology

But even when a full semantic model is not available, it may still be possible to help a person draft a text by guessing what she intends to write based on what she has already keyed in. The idea of automatically extending, or completing the tokens (if not the full text) that a user is typing has a long history that goes back to the earliest UNIX systems, which included both command-name and file-name completion. The success of the operation here clearly depends on there being a relatively small number of items for the completion program to select among, exploiting a very simple heuristic: use the list of available command names when typing at the start of the line and the list of available filenames otherwise. This principle also applies to interactive development environments (e.g. Eclipse and Visual Studio) or XML editors (e.g. oXygen and XML Spy), which suggest correct completions given the context of the piece of text the user is drafting. The semantic model in this case is more complex because it involves the parsing of both the program text the user is writing, as well as previous program parts. This is possible because both the text and its environment are defined in the context of artificial languages which have a rigid grammar and a well-defined semantics.

The widespread popularity of SMS (short message services) on mobile phones constitutes another, extremely successful application of predictive text technology. Owing to the telephone's limited keypad, every sequence of key presses that is meant to convey a word is necessarily ambiguous; and here, the great leap forward came with the addition of a monolingual dictionary and statistical language models, which together allow many of the words in the message to be automatically disambiguated. In fact, today's SMS systems can often complete a word before the user has finished typing it, and they will reorder the completions in the suggestion box with each new key the user enters. Some systems now propose multiple word expressions and others can even acquire the user's preferred vocabulary from past messages or files.

Predictive text technology is also used to assist people with various types of learning disabilities or those who, for whatever reason, feel intimidated or lack the requisite language skills to compose a written text (even in their native tongue). For example, the Canadian company Quillsoft has developed the WordQ program, a plug-in that inserts a small pop-up in a word processing or text editing window in which completions are suggested for the current word; and after each word, or at the end of the sentence, the software uses speech synthesis to read the completed unit back to the user. For children with impaired writing skills whose spoken vocabulary far exceeds their written vocabulary, this kind of automated word completion followed by synthesized read-back helps to reinforce links between the two forms of language. At the other end of the disability spectrum, there are augmentative communication systems designed for people with severe motor handicaps, for whom every keystroke demands considerable physical effort. For example, the Sibylle system (Wandmacher et al., 2007) offers two types of assistance to its users: an on-screen keyboard, which is dynamically reconfigured so as minimize the effort required to select the next letter of a word; and another completion window in which predictions of the following word are proposed on the basis of those that have preceded. Because spelling and other input errors are common in this particular context, Sibylle's word completion system is not based on a standard frequential dictionary, like the other word

[2] For any given region in Canada, a new public weather bulletin is published four times a day, every day of the year. Documentation on a pharmaceutical product will normally be published once and will last for the life of the product.

completion systems discussed so far, but rather on an n-gram model that operates on the level of characters; and a similar model is used on the sentence level to calculate the word predictions.

4. Machine translation

What distinguishes the process of translation from other acts of writing is that the translator works simultaneously with not one but two texts: a source text in one language and a target text that he is drafting in a second language. A central part of the translator's job is to ensure the semantic equivalence of these two texts. However, the availability of the source text makes it possible to envisage novel kinds of authoring aids that are not possible in other, monolingual contexts.

As we previously noted, fully automatic machine translation is still not capable of systematically producing publication-quality translations. In fact, until recently, MT output was not generally considered revisable by most translators, but only suited for gisting purposes. Machine-aided human translation (MAHT) eliminates the MT component, leaving the translation to the human translator, while attempting to provide her with reliable tools that will hopefully increase productivity. However, surprisingly few types of aids have been proposed for MAHT thus far: basically, tools for terminology management and repetitions processors (more commonly known as translation memories). The problem with the former is that its impact on productivity is minimal; the problem with the latter is that the great majority of texts do not contain a sufficient level of full-sentence repetition to make them cost-efficient.

For some years now, the RALI laboratory has been developing the TransType system (Foster et al., 1997; Langlais et al., 2004), a novel approach to translation automation that aims to strike a compromise between MAHT and fully automatic MT. In contrast to MAHT, it retains an MT component; it thus has a greater potential to increase translator productivity than current MAHT tools. Unlike fully automatic MT, however, it doesn't allow the system to operate with complete autonomy, but instead seeks to conjugate the power of the machine with the discernment of the human translator, in a way that allows the two to productively collaborate in the text drafting process.

TransType represents a new kind of interactive MT in which the focus of the interaction is on the target text, unlike classic interactive systems, where the aim of the human intervention is to disambiguate the source text; c.f. (Kay, 1973). In TransType, a statistical MT system is embedded within an interactive editing environment. The MT system's contributions to the target text take the form of predicted extensions to the current linguistic unit, based upon its automatic translation of the source text and a target language model. (See Figure 1 at the end of this paper for a snapshot of a TransType session.) TransType is intended to be used by a professional translator whose contribution to this joint undertaking is to serve as guarantor of the quality of the target text being drafted; as such, she must evaluate the system's proposed extensions and either accept or reject them, in whole or in part. To reject a proposed completion, the translator simply continues typing the target text she has in mind. However, TransType can immediately take advantage of this additional, reliable information on the desired target in order to propose an alternative (and hopefully better) completion, which the user may again accept or reject; and so on, until a fully acceptable translation has been composed. Obviously, the more accurate the system's predicted completions, the less the user will have to intervene, which in principle should translate into increased productivity gains. For more on the results of extensive user trials with TransType, see (Macklovitch, 2006).

A major advantage of this interactive-predictive approach is that it actively involves the human translator in the text generation process, doing that for which she is most qualified, i.e. translating, as opposed to performing linguistic analyses, or post-editing (i.e. correcting) raw MT output. TransType thus avoids one of the principal objections that translators have long harboured against MT, to wit, that it places them in a role of subservience to a machine whose comprehension of natural language is considerably less than theirs.

5. Drafting in a Foreign Language

Let us now return to the situation where no meaning representation is available and where, as we have seen, current authoring aids can offer little more than simple word completion. Needless to say, for people with serious learning disabilities or physical handicaps, there is absolutely nothing wrong with word completion; on the contrary, it can appreciably facilitate text entry for them. However, in the absence of such obstacles, word completion can be rather frustrating for an "ordinary" person who is attempting to draft a text. Simply put, the time and effort required to evaluate the list of proposed completions usually outweigh the benefit of not having to type a few characters. This at least is the case for most moderately literate monolinguals.[3]

But what about people who are called upon to draft texts in a second or a foreign language? In the domain of scientific research, for example, English has become the lingua franca for international conferences and scientific publications and, as a consequence, many researchers whose mother tongue is not English are obliged to write technical articles in that language. It is our contention that authors in this situation might be more tolerant of word completion than native English speakers. At the very least, it could help them avoid certain errors of spelling and grammar during the text drafting process.

What sorts of errors do authors in this situation typically commit? (And again, we're focusing here on people who may have considerable competence in the foreign language, although this will tend to be more passive than active competence, i.e. it derives principally from reading texts in their domain of technical specialization.) An Anglophone drafting in French, for example, will often make errors of gender agreement, either selecting the incorrect article for a masculine or feminine noun, or

[3] Contrary to what is suggested in the CFP for this workshop, ordinary text composition does not strike us as an undertaking that is generally fraught with peril. Writing is certainly a complex task, but educated adults are not often overwhelmed by these complexities, and when they do lose their orientation, they usually manage to recover it. Which is not to say that the texts they author are always stellar or models of clarity.

failing to properly inflect a noun modifier. Such errors arise, of course, because nouns are not marked for gender in English. Another major difference between the two languages is the high degree of inflection in the French verbal system; this too will result in Anglophones committing not just more errors in their French texts, but also different types of errors than those found in the texts of Francophones. On the other hand, a Francophone drafting in English will frequently have trouble with the null article, which English uses to denote generic reference (among other things). Since French does not generally allow nouns with no article, Francophones will tend to insert one where no English speaker ever would, e.g. *'she loves the classical music'*. Both Anglophones and Francophones will often err in their selection of governed prepositions, since these can be wholly unpredictable and do not directly translate from one language to the other. So, for example, no Anglophone would ever use the preposition 'of' to introduce the complement of the verb 'depend', although one could well find this in a text drafted by a Francophone; and conversely, no self-respecting Francophone would ever write 'dépendre sur' (unless under the pernicious influence of an English calque). Note that some of these errors might be detected *a posteriori* by a sophisticated monolingual grammar checker.[4] What we're suggesting here, however, is that it might be more convenient for people drafting in a foreign language to have a specialized foreign language drafting tool that would hopefully avoid the generation of such errors in the first place.

Such a system would naturally be more helpful if it were able to go beyond simple word completion and propose multi-word or even full-sentence completions, as the TransType system does. In fact, a series of extended user trials with TransType led us to fundamentally modify our approach to text completion. Instead of systematically predicting a fixed amount of text after each character the user types, later versions of the system attempted to make predictions that would maximize the expected benefit to the user in each context; c.f. (Foster et al., 2002). This expected benefit was estimated from two components: a statistical translation model which calculated the probability that a predicted completion would be correct or incorrect, and a user model which factored in (among other things) the time required to read a more or less lengthy prediction. In addition, later versions of TransType allowed the user to adjust its predictions to her preferences and taste, e.g. by setting a confidence threshold, or fixing a minimal length for a completion, or a maximum number of alternate completions. Moreover, if on a given text the system's predictions were wholly unsatisfactory (for whatever reason), the user could always turn off the prediction engine and type the translation on her own. We suggest that this same kind of flexibility should be retained for monolingual authoring aids that attempt to suggest completions that extend beyond the current word.

Of course, the crucial question that remains is this: In the absence of an underlying meaning representation or a parallel text in another language, what would be the source of such multi-word completions? How likely is it that general-purpose language models trained on large, publicly available corpora (of journalese or parliamentary debates, for example) will be able to propose extended completions that match an author's intentions? Not very likely, in our view. Our experience with TransType suggests that the predictions would be too diffuse and would require too many interactions before they manage to zero in on the sentence the author has in mind. However, the situation that we evoked above, i.e. of a person drafting a specialized text in a foreign language, is slightly different and would not require that the predictor rely exclusively on general-purpose texts for its training material. Here, ideally, we would prefer to train the system's language model on a corpus composed of native speakers' texts taken from the same specialized domain that the author is drafting in. Moreover, our foreign language drafting tool could perhaps take a cue from some of the simpler word completion systems described above and allow the user to explicitly select certain multi-word expressions in this corpus so that they are added to the system's dictionary. In the domain of computational linguistics, for example, a non-English speaking user might flag such multi-word terms as 'cross-language information retrieval' or 'statistical machine translation', thereby allowing the system to propose their completion after only a few characters.

Another potential advantage of this interactive-predictive approach is that it could *learn* from its interactions with the user, i.e. from the completions the user has accepted and those that have been rejected, in order to improve its underlying model. Most word completion systems adapt their predictions to new input from the user, to ensure their compatibility with the input prefix. However, what we're proposing here is not simply adapting the predictions of a fixed model, but rather updating the parameters of that model on the fly so that it would actually improve with use. We have done some initial experiments in this vein (Nepveu, 2004) using a cached language model with the TransType system, and the gains, though modest, were encouraging.

In short, this kind of interactive text prediction would seem to offer interesting ways of harnessing the power of language models for the purposes of foreign language drafting. Would the additional restrictions that come with this application – i.e. highly targeted training material and a more tolerant author drafting in a second language – suffice to make an interactive-predictive text completion system beneficial for such a user? In particular, would the specialized language models allow for accurate multi-word or phrasal predictions? It is difficult to say without actually implementing the system and then carrying out tests. However, we find the idea sufficiently intriguing to want to undertake the development of such a system. Anyone care to join us?

[4] Both Antidote and the grammar checker in Word, for example, are quite good at detecting errors of agreement in French, although their performance is less impressive on governed prepositions.

6. References

Chandioux, J. & Guéraud, M-F. (1981). MÉTÉO : un système à l'épreuve du temps. *META,* 26(1), pp. 18--22.

Goldberg, E., Driedgar, N., Kittredge, R. (1994). Using natural-language processing to produce weather forecasts. *IEEE Expert,* 9(2), pp. 45--53.

Foster, G., Isabelle, P., Plamondon, P. (1997). Target-Text Mediated Interactive Machine Translation. *Machine Translation*, 12(1-2), pp. 175--194.

Foster, G., Langlais, P., Lapalme, G. (2002). User-Friendly Text Prediction for Translators. In Proceedings of *EMNLP 2002*, Philadelphia, PA., pp. 148--155.

Hartley, A. & Paris, C. (1997). Multilingual Document Production: From Support for Translating to Support for Authoring. *Machine Translation,* 12(1-2), pp.109--128.

Kay, M. (1973). The MIND System. In R. Rustin (ed.) *Natural Language Processing*, New York, N.Y.: Algorithmics Press, pp. 155--188.

Langlais, P., Lapalme, G., Loranger, M. (2004). TransType: Development-Evaluation Cycles to Boost Translator's Productivity. *Machine Translation* (Special Issue on Embedded Machine Translation Systems), 17 (2), pp. 77--98.

Lapalme, G., Brun, C., Dymetman, M. (2003). MDA-XML : une expérience de rédaction contrôlée multilingue basée sur XML. In Proceedings of *TALN 2003*, Batz-sur-Mer, France, pp. 379--384.

Macklovitch, E. (2006). TransType2: The Last Word. In Proceedings of *LREC-2006*, Genoa, Italy pp. 167--172.

Nepveu, L. (2004). Adaptation de modèles de traduction dans le cadre du projet TransType. Mémoire de maîtrise en Informatique, Université de Montréal.

Power, R. & Scott, D. (1998). Multilingual authoring using feedback texts. In Proceedings of *COLING-ACL 1998*, Montreal, Canada, pp. 1053--1059.

Reiter, E. & Dale, R. (1997). Building applied natural language generation systems. *Natural Language Engineering,* 3(1), pp. 57--87.

Wandmacher, T., Béchet, N., Barhoumi, Z., Poirier, F., Antoine, J-Y. (2007). Système Sibylle d'aide à la communication pour personnes handicappés. In Proceedings of *TALN 2007*, Toulouse, France, pp.539--548.

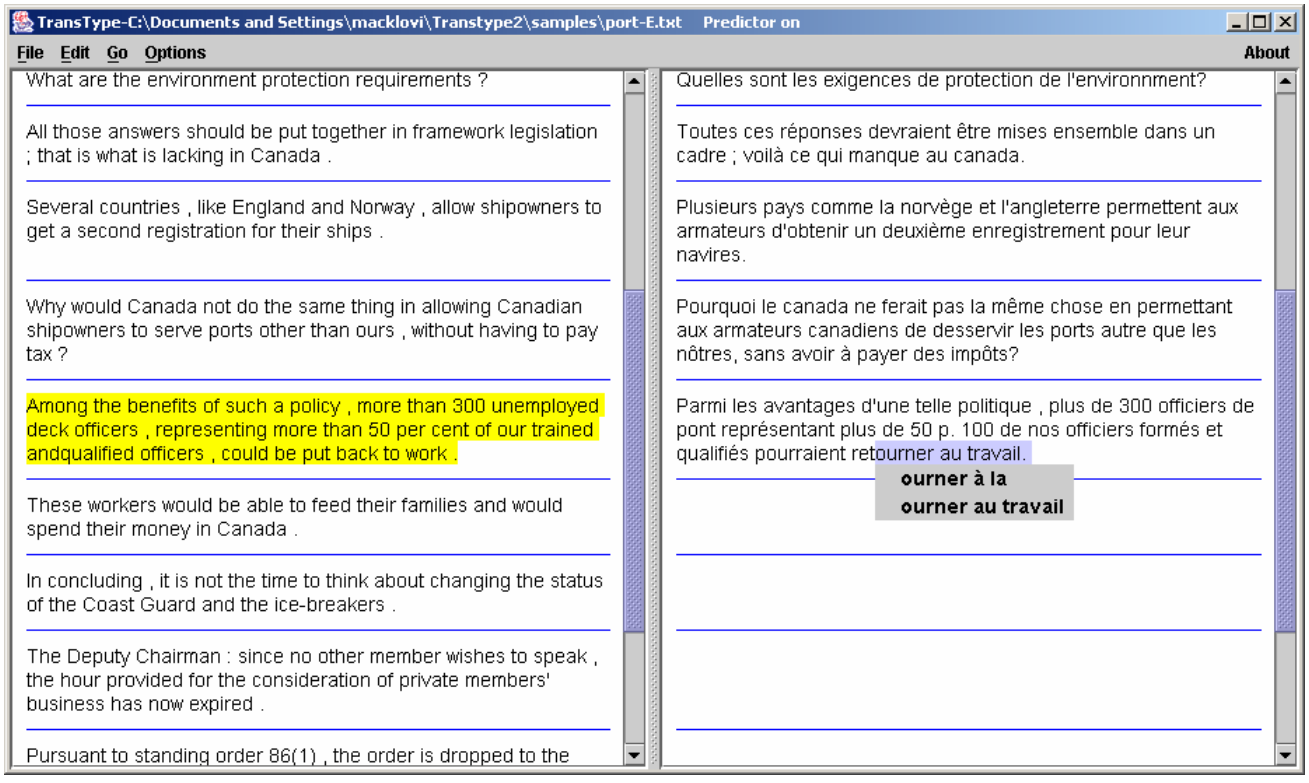

Figure 1 : Snapshot of TransType session

The English source text appears in the left pane; on the right is the French translation currently being drafted. To assist the human translator, TransType proposes completions to the target translation, generated on the fly, using a statistical translation model combined with target language model. Here, three completions are suggested: to accept the top one, the translator need only hit the Enter key; the others can be inserted into the text, in whole or in part, by pointing and clicking with the mouse. Other versions of TransType are capable of proposing longer predictions, up to full-sentence length.

Language-Aware Text Editing

Cerstin Mahlow, Michael Piotrowski, Michael Hess

Institute of Computational Linguistics, University of Zurich
Binzmühlestrasse 14, 8050 Zürich, Switzerland
{mahlow, mxp, hess}@cl.uzh.ch

Abstract

While software developers have various "power tools" at their disposal that make the writing of computer programs more efficient, authors of texts do not have the support of such tools. Text processors still operate on the level of characters and strings rather than on the level of word forms and grammatical constructions. This forces authors to constantly switch between low-level, character oriented, editing operations and high-level, conceptual, verbalisation processes. We suggest the development of language-aware text editing tools that simplify certain frequent, yet complex editing operations by defining them on the level of linguistic units. Pluralizing an entire noun phrase plus the verb forms governed by it would be an ambitious example, swapping the elements of a conjunctive construction a more modest one.

We propose a taxonomy for revising and editing operations with respect to revising and editing as such as well as engineering desiderata. We describe the components of a pilot implementation for German where these operations are seamlessly integrated with the standard functions of an existing open-source editor. The operations can be invoked on demand and do not intrude on the authoring process. Changes can be performed locally or globally, thus simplifying the writing process considerably, and making the resulting texts more consistent.

1. Introduction

The process of creating a high-quality text involves several iterations of writing, revising and editing. Ever since the beginnings of computerized word processing in the 1960s (see Haigh (2006) for a historical overview) researchers tried to develop tools to support writers. In the history of text processing aids there were several attempts (see, e.g., Oakman (1994)) to create expert tools for expert writers, but today's word processors mainly operate on the level of characters rather than linguistic units. The result are typical revision and editing errors such as duplicate verbs or extraneous conjunctions.

In this paper we propose functions for word processors that aim at improving the "brain-to-hand-to-keyboard-to-screen-connection" (Taylor, 1987, p. 79). We will show that editors can be upgraded to provide functions that operate on linguistic elements with relatively low costs in terms of linguistic resources. We will outline what aspects need to be considered when implementing these functions.

2. Motivation

We think that many of the revision and editing errors are caused by *attentional disruption* since writers have to translate their high-level goals (e.g., changing the grammatical mood of an expression) into the low-level, character-oriented functions of the editor.

Even simple revisions, such as changing "editing and revising" to "revising and editing", which *conceptually* merely consists of swapping the conjuncts, require substantial planning and memory capacity when they have to be executed in an editor or word processor. [1]

However, as McCutchen (1996) points out, the capacity of a writer's working memory and, of course, cognitive resources in general are limited, and when resources are diverted to other activities, this will have a negative impact on writing processes (i.e., planning, translating, and reviewing).

In this paper we will concentrate on texts in German. German is an interesting language for our research because, as a heavily inflectional language, it is morphologically much richer than English, and this has implications for revising and editing. For example, globally replacing one word with another, say, *hut* with *tent*, doesn't pose any problem in English, since there are only three word forms and no changes to the stems. Apart from a very small number of exceptions (such as *mouse* or *foot*), all replacements can thus be done by simple regular expressions. Inflectional languages typically also have much freer word order than English so that writers are more likely to change the syntactic structure during revision, but moving some words may require adjustments in other parts of the sentence.

Our target group are *experienced writers*, i.e., the functions we are proposing are intended for supporting writers who know how to write and who are used to expressing their writing and revising actions in exact terms. When writers think and talk about texts, they do this on the level of *linguistic units*, not on the level of *characters*, even though they don't necessarily use linguistic *terms*, such as *genus verbi*. So, for example, they may say, "Let's put this sentence into passive voice". Other examples may be "This phrase should be plural", "This sentence is too long, break it up", "Merge these two sentences", or "We should use 'laptop' instead of 'personal computer"'.

3. Improving word processing – related work and state of the art

Even early editors had functions operating on other units than characters and lines, viz. "words" and "sentences" (cf. van Dam and Rice (1971), Callender (1982)). These units were, however, only defined in terms of the writing system, i.e., a "word" was defined as a string bounded by whitespace

[1] See (Mahlow and Piotrowski, 2008, p. 639) for a detailed description.

or punctuation, and a "sentence" was defined as a string terminated by a period followed by two space characters.
There has been research on automatic spelling, grammar, and style checking at universities and research institutions ever since computers started being used for writing natural-language text; Cherry (1981) provides an overview of some early research. With the spread of commercial word processors with graphical user interfaces and integrated checkers most of these research systems disappeared (cf. Vernon (2000, p. 332)). After several years of using commercial software for composing – and for teaching how to compose – the need for more suitable functionalities arose again, along with criticism of the functionality of word processors and checkers for spelling, grammar, and style (see, for example, Piolat (1991), Dale (1997), Vernon (2000), McGee and Ericsson (2002)). But today, more than one decade later, we are still lacking better functions. Popular word processors still operate on non-linguistic entities and checkers for spelling, grammar, or style are still unreliable.
Today, word processors, whether commercial or open-source, whether used for print or for the Web, all have very similar graphical user interfaces. You will find nearly identical buttons and menus for nearly identical sets of functions – setting text in italics, changing the text color, switching from flush left to fully justified alignment, etc. Writers are accustomed to these functions and to this feature set. Sharples and Pemberton argued in 1990 already that writers "can infer the existence of operations and how to perform them from previous experiences" (Sharples and Pemberton, 1990, p. 49). At this time, the use of word processors was not as common as it is today, and today's writers are even more used to core operations as *cut*, *copy*, *paste*, etc. Since users are conditioned on the core operations and their behavior, experimental editors would have to behave in the same way to be accepted.
Research projects that proposed new and improved functionality by designing completely new editors with a completely different look and feel (see, for example, Dale (1990; Williams (1990; Holt et al. (1990; Sharples and Pemberton (1990; Dale (1997)) didn't have much success, and certainly no lasting impact. We thus think writers have to be offered new and – in our opinion – more suitable functions for professional writing, revising and editing as an *extension* to their preferred word processor or editor. In the 1990s, performance issues may also have been a motivation for the creation of new editors, but we now have enough computing power for on-the-fly analyses and operations, and we can thus take a different approach.

4. Approach

Writers are used to the functions that their editor of choice provides. Having the possibility to directly compare the new functions with the previous way of performing an operation should promote the acceptance of the new functions and help with the evaluation. We are therefore implementing new functions in the XEmacs[2] editor, which will thus serve as a test bed for testing new functions with *real* users doing their daily work. They will still be able to use the existing functions and will not have to learn how to use a new editor.

[2] `http://xemacs.org/`

We do not want to create a new editor, and we are not interested in developing another grammar checker (or some other postwriting tool) either. As Sharples (1999, p. 190) points out, advanced writers tend to turn off this type of "assistance", or they try to ignore it (McGee and Ericsson, 2002, p. 462). The problem with checkers is that they are good for ensuring consistency; advanced writers, however, are interested in creating specific effects (whether in terms of vocabulary, syntax, or style) to achieve their specific communicative goals. Advanced writers know what they are doing and they know why they are doing it. We do not want to say that checkers are useless: They are certainly good tools for detecting possible errors and for ensuring consistency. However, for advanced writers, these tasks come typically very late in the writing process (in the postwriting phase) and are just another proofing step.
What we are more interested in is adding *language awareness* (as in editors for programming languages) and creating functions that support writers *during* the revision and editing process by simplifying complex operations. Each of these functions should assist the writer in one specific editing task that is tedious or error-prone to carry out manually. The support must be designed in a way that makes it a better alternative. Thus, it must not interfere with the writing process and it must not make any assumptions about what the writer *might* want to do, or whether this is "correct", but it must perform the task quickly, *reliably* (another problem with checkers is that they are not reliable), and under the control of the writer. Also, as Williams (1990, p. 7) points out, the functions have to be interactive to be useful during the process of composing.

4.1. Types of functions

Generally speaking, language awareness can result in two types of functions:

Informational functions Elements – such as words, phrases, or clauses – can be highlighted (known as *syntax highlighting* in programming editors), or the writer can request information about certain aspects of the text, such as prepositions used, conjuncts, sentences without verbs, or variants of multi-word expressions. The writer has to interpret the results himself and can decide how to make use of them.

Operations The other type are functions that operate on and make changes to textual elements. Linguistic elements can be reordered, modified, or deleted. In order to reduce the cognitive load, it is desirable to reduce the number of actions necessary to reach a specific goal (Allen and Scerbo, 1983). This can be achieved by combining sequences of core operations into higher-level functions closer to the users' goals and their mental model of the task. These functions always behave in the same way and none of the involved core operations will be forgotten or executed twice. The cost of the "brain-to-hand-to-keyboard-to-screen" process is thus reduced.

For both types of functions we can find examples requiring only minimal linguistic knowledge. Functions requiring

more linguistic knowledge differ with respect to the required resources: In some cases, only static resources, e.g., a list of all conjunctions of a language, are necessary. In other cases, morphologic analysis and/or generation may be needed at run time. Sometimes an operation may even need syntactic knowledge about the context. Since sufficient computing power is available today to perform linguistic analyses on the fly, and since linguistic resources, such as morphologic components, are available for different languages at a reasonable cost or even as open source, it is now realistic to employ them in an interactive editing environment.

For example, highlighting conjunctions may help writers in getting an overview of their argumentation structure. Executing the command `show-conjunctions` would give a quick overview of the use of conjunctions. To implement this function, only minimal linguistic resources are required. As conjunctions are typically invariable, there is no need to look for different word forms, and since conjunctions are not linguistically productive (i.e., no new conjunctions are produced using derivation or composition), it is not necessary to consider morphological processes. Thus, only a list of conjunctions is needed.

Detecting sentences without verbs is a more complex example of an informational function. Here, linguistic resources are clearly necessary: First, sentence boundaries must be identified, and then the POS of the word forms of each sentence have to be determined. Thus, this can be considered a more advanced function.

For operations, swapping conjuncts may serve as an example requiring only minimal linguistic resources. This function, as mentioned in section 2., requires certain core operations, depending on the editor. This type of function is aware of (and dependent on) the writing system but needs no further linguistic knowledge, e.g., of the word classes involved – and we do not want to restrict writers in what they can transpose.

At the other end of the spectrum would be a function for replacing words, i.e., all word forms of a word should be replaced with the corresponding word form of another word. However, in inflectional languages, like German, words can have many word forms and each word form can typically express more than one category (see table 1 for the paradigms of two German nouns).

Manually replacing all occurrences of *Zelt* ‘tent’ with the corresponding word form of *Haus* ‘house’ is therefore a complex task: First, one has to find all word forms of *Zelt* – with the usual search functions this will require to search for each word form individually. Then, one has to determine the category of a specific occurrence; note that the word form may be ambiguous, and the exact category can only be found by looking at the syntactic context. Finally, one must manually replace the word form of *Zelt* with the corresponding word form of *Haus*. [3]

We are thus proposing a function `query-replace-word`, which would operate as follows: After calling the function, the writer is prompted to enter the word to replace (*from-word*) and its replacement (*to-word*). The function searches for all forms of the paradigm of *from-word*; when a form of *from-word* is found, it is replaced with the corresponding form of *to-word*. It is clear that this task requires morphological analysis and generation. In fact, replacing the word form *Zelte* with the corresponding word form of *Haus* requires even syntactical analysis: *Zelte* can be of the categories DatSg, NomPl, GenPl, AccPl, but *Haus* has different word forms for DatSg on the one hand and for NomPl, GenPl, AccPl on the other hand.

Word	**Forms**	**Categories**
Zelt	Zelt	NomSg,
(*n*, (e)s/e decl.)		DatSg, AccSg
	Zeltes	GenSg
	Zelts	GenSg
	Zelte	DatSg, NomPl,
		GenPl, AccPl
	Zelten	DatPl
Haus	Haus	NomSg,
(*n*, (e)s/er decl.)		DatSg, AccSg
	Hauses	GenSg
	Hause	DatSg
	Häuser	NomPl, GenPl,
		AccPl
	Häusern	DatPl

Table 1: Word forms of *Zelt* and *Haus*.

4.2. Components of a pilot implementation

We have chosen XEmacs as an implementation testbed for the following reasons: It is open-source and new functions can easily be added using Emacs Lisp, either as additional functions or replacing existing functions.

All functions – informational functions as well as operations – will be implemented in a new minor mode called *natlang-mode*. This mode can then be used with various major modes, such as *LaTeX-mode*, *text-mode*, or *message-mode*, i.e., in all modes dealing with natural-language texts. The syntax highlighting facilities can be used for implementing various informational functions.

As usual in XEmacs, functions can either be bound to keystrokes, e.g., `C-c C-r` for `transpose-conjuncts`, or by called by name, e.g., using `M-x transpose-conjuncts`. Functions can also be called by selecting them from a menu. Since these functions will be implemented in a similiar way existing functions are implemented, optional or required parameters can be specified in the usual ways.

We make use of the internal handling of XEmacs for words and sentences: A “word” is a character string bounded by spaces, a “sentence” begins with a capitalized word and ends with a period. To distinguish this period from the ones in abbreviations in an easy way, we require writers to follow sentence-final periods by two spaces, as it is commonly done in English, even though this practice is not normally used in German. Of course, in a more advanced implementation we will use a proper sentence-detection component. By considering spelling German conventions, we can also extract some basic information from the text: Nouns and proper names are always capitalized, while all other word classes

[3] For a discussion of side effects of such functions see (Mahlow and Piotrowski, 2008).

start with a lower case letter except when at the beginning of a sentence.
The morphological component we use for German is DMM (Deutsche Malaga-Morphologie) (Lorenz, 1996).

5. Towards a taxonomy of language-aware functions

In section 4.1. we have presented some first thoughts on linguistic support for revising and editing. However, we still need to get a better idea of what is actually happening when writers are revising and editing. The processes can be described with respect to various dimensions, some of which are:

Destructiveness As described in section 4.1., we distinguish between *informational functions* and *operations*. Informational functions highlight elements to help the writer in finding certain occurences of a linguistic element or phenomenon. They show results of certain analyses, e.g., the number of sentences lacking a finite verb or variants of multi-word expressions. Operations modify linguistic elements such as words, phrases, or clauses. *Destructivness* is a boolean attribute.

Level of language dependence The type of the required linguistic resources determines the level of language dependence of a function.

For functions like `transpose-conjuncts` it is sufficient to use the core operations of a word processor and the properties of the writing system of a certain language. It is not necessary to analyze the words affected by this function, and the definition of a *word* as a string bounded by whitespace or punctuation is completely sufficient for functions that transpose words or move the cursor. Thus, functions using basic concepts and principles of a certain language and core operations will have a low level of language dependence.

Functions like `show-conjunctions` have a higher level of language dependence: Linguistic resources are required, in this case a list of conjunctions. In general, highlighting functions, or functions operating on invariable elements, need similar linguistic resources, i.e., word lists or lexica, but they do not need explicit morphological or syntactical rules. We call these linguistic resources *static*.

The highest level of language dependence includes functions like `query-replace-word`. Here, morphological analysis and generation and, in some cases, even syntactic analysis are involved. The linguistic resources have to be in a form suitable for run-time execution. We call these linguistic resources *dynamic*.

The *level of language dependence* is not an absolute value. It can only be determined by comparison with the requirements of other functions.

Complexity We measure *complexity* in terms of core operations required. The value for *complexity* can be specified as an absolute number, or it can be specified on a scale between *high* and *low* by comparing it with the corresponding values of other functions.

We have to take into account that a certain function can be split into *core operations*, i.e., atomic editor operations, or into *involved operations*. If we combine some of our additional functions, each consisting of a certain sequence of core operations, into even more complex functions, the constituent functions will be called *involved operations*.

Specificity The kind of arguments a function takes determines the *specificity* of the function. Some functions take a *concrete element*, such as a word like "house", while others operate on *classes* like conjunctions, finite verbs in preterite tense, or relative clauses. The value for *specificity* is an element from a set of argument types.

Area of operation We use this term to describe whether a function operates on one specific occurrence of a linguistic element or whether it operates on all occurrences of an element. For example, a function such as `transpose-conjuncts` only operates at the current cursor position, whereas a function such as `query-replace-word` is designed to operate on all occurrences of the specified *from-word* (even if the user chooses to interrupt the function at some point). The *area of operation* is either *local* or *global*.

Side effects When executing an operation it can happen that the result is not grammatical. This is clearly a unintended *side effect*. Such an operation will thus require further editing operations on other elements to restore grammaticality. *Side effects* can be of different types, such as agreement errors or dangling anaphoric references. *Side effects* can also occur at different distances from the original operation – they may be restricted to the same phrase (e.g., adjective-noun agreement), or they may occur further away, even in other sentences.

We can distinguish the values *has side effects* and *has no side effects*. For operations with side effects we can determine the type of side effects as one element from a set of types.

To be able to determine these dimensions it is necessary to analyze and to classify user actions, e.g., from keystroke recordings (cf. Good (1985), Flinn (1987), Perrin (2006)), and to consider linguistic rules and engineering issues.
The classification can serve as a help in making decisions, and it can then be used to write specifications and serve as a guideline for the actual implementation.

6. Conclusion and further work

We think that language-aware editing functions can relieve writers from many low-level operations which distract from the actual revision and editing process. Our approach concentrates on support *during* the writing process, enabling the writer to interact with the word processor.
We showed some examples for such functions, concentrating on German. These functions can be seen as add-ons to existing word processors, allowing writers to use the functionality they are accustomed to and benefit from the new ones. The required resources are relativly small but

can have a considerable impact on the writing process. We then outlined a number of aspects which have to be taken into consideration, and which should eventually result in a taxonomy of revising and editing operations.

Currently we are selecting the operations to be implemented according to the principles described in this paper. We will then evaluate them with experienced writers.

Once we have an implementation serving as a proof of concept, we will then be able to consider additional aspects: What is the best way to make writers learn new functions? What is the best way to call these functions: using keystrokes or using pull-down menus? Which linguistic *terms* do writers really use when talking about linguistic *units*, and which terms should be used in editor functions?

Acknowledgements We thank Yves Forkl and Rolf Schwitter for fruitful discussions.

7. References

Robert B. Allen and M. W. Scerbo. 1983. Details of command-language keystrokes. *ACM Trans. Inf. Syst.*, 1(2):159–178, April.

E. D. Callender. 1982. An evaluation of the AUGMENT system. In *SIGDOC '82: Proceedings of the 1st annual international conference on Systems documentation*, pages 29–35, New York, NY. ACM Press.

Lorinda Cherry. 1981. Computer aids for writers. *ACM SIGOA Newsletter*, 2(1-2):61–67.

Robert Dale. 1990. A rule-based approach to computer-assisted copy-editing. *Computer Assisted Language Learning*, 2(1):59–67.

Robert Dale. 1997. Computer assistance in text creation and editing. In Giovanni B. Varile, Antonio Zamponelli, Ronald Cole, Joseph Mariani, Hans Uszkoreit, Annie Zaenen, and Victor Zue, editors, *Survey of the State of the Art in Human Language Technology*, pages 235–237. Cambridge University Press, New York, NY.

Jane Z. Flinn. 1987. Case studies of revision aided by keystroke recording and replaying software. *Computers and Composition*, 5(1):31–44, November.

Michael Good. 1985. The use of logging data in the design of a new text editor. *SIGCHI Bull.*, 16(4):93–97, April.

Thomas Haigh. 2006. Remembering the office of the future: The origins of word processing and office automation. *Annals of the History of Computing, IEEE*, 28(4):6–31.

Patrik O. Holt, Dag C. Hegg, and Terje Johnsen. 1990. Engineering written style. *Computer Assisted Language Learning*, 2(1):27–35.

Oliver Lorenz. 1996. Automatische Wortformerkennung für das Deutsche im Rahmen von MALAGA. Master's thesis, Friedrich-Alexander-Universität Erlangen-Nürnberg.

Cerstin Mahlow and Michael Piotrowski. 2008. Linguistic support for revising and editing. In Alexander Gelbukh, editor, *Computational Linguistics and Intelligent Text Processing: 9th International Conference, CICLing 2008, Haifa, Israel, February 17–23, 2008. Proceedings*, pages 631–642, Heidelberg. Springer.

Deborah McCutchen. 1996. A capacity theory of writing: Working memory in composition. *Educational Psychology Review*, 8(3):299–325.

Tim McGee and Patricia Ericsson. 2002. The politics of the program: MS Word as the invisible grammarian. *Computers and Composition*, 19(4):453–470, December.

R. L. Oakman. 1994. The evolution of intelligent writing assistants: trends and future prospects. In *Tools with Artificial Intelligence, 1994. Proceedings., Sixth International Conference on*, pages 233–234.

Daniel Perrin. 2006. Schreibforschung im Kursalltag: Was die Progressionsanalyse praktisch nützt. In Otto Kruse, Katja Berger, and Marianne Ulmi, editors, *Prozessorientierte Schreibdidaktik: Schreibtraining für Schule, Studium und Beruf*, pages 279–294. Haupt Verlag, Bern, Stuttgart, Wien.

Annie Piolat. 1991. Effects of word processing on text revision. *Language and Education*, 5(4):255–272.

Mike Sharples and Lyn Pemberton. 1990. Starting from the writer: Guidelines for the design of user-centred document processors. *Computer Assisted Language Learning*, 2(1):37–57.

Mike Sharples. 1999. *How We Write: Writing As Creative Design*. Routledge, June.

Lee R. Taylor. 1987. Software views: A fistful of word-processing programs. *Computers and Composition*, 5(1):79–90.

Andries van Dam and David E. Rice. 1971. On-line text editing: A survey. *ACM Comput. Surv.*, 3(3):93–114, September.

Alex Vernon. 2000. Computerized grammar checkers 2000: capabilities, limitations, and pedagogical possibilities. *Computers and Composition*, 17(3):329–349, December.

Noel Williams. 1990. Writers' problems and computer solutions. *Computer Assisted Language Learning*, 2(1):5–25.

Enhancing the OpenOffice.org Word Processor with Natural Language Processing Capabilities

Thomas Gitzinger[1] **and René Witte**[2]

[1]Institut für Programmstrukturen und Datenorganisation (IPD)
Universität Karlsruhe (TH), Germany

[2]Department of Computer Science and Software Engineering
Concordia University, Montréal, Canada
http://semanticsoftware.info

Abstract

Today's knowledger workers are often overwhelmed by the vast amount of readily available natural language documents that are potentially relevant for a given task. Natural language processing (NLP) and text mining techniques can deliver automated analysis support, but they are often not integrated into commonly used desktop clients, such as word processors. We present a plug-in for the OpenOffice.org word processor *Writer* that allows to access any kind of NLP analysis service mediated through a service-oriented architecture. *Semantic Assistants* can now provide services such as information extraction, question-answering, index generation, or automatic summarization directly within an end user's application.

1. Introduction

Information Retrieval (IR) is, in some respect, a solved problem: Users nowadays have immediate access to vast amounts of information. Popular search engines, such as *Google*, can deliver more documents in a fraction of a second than any human can process in a lifetime. As (Simon, 1971) pointed out, "A wealth of information creates a poverty of attention," and this statement certainly fits the information age. Practically no one dealing with large amounts of reports, literature, drafts, articles and the like, can read everything they would like to. This becomes a problem when decisions have to be made and not all the information that is theoretically available can be taken into account. It becomes a problem when an expert or a reporter must gain an overview of a large corpus of literature, be it news articles, opinion pieces or technical reports, and has a very limited time frame.

While one might argue that retrieval speed and precision of IR can still be improved, we believe the most important advances in the near future will focus on improving the automatic processing of retrieved information, thereby allowing the human user to gain time for his actual task of evaluating the information and creating new value from it. But although natural language processing (NLP) and text mining research has made impressive progress over the last decade, the developed technologies have not yet found wide adoption in end-user tools commonly used for reading and developing content. Knowledge workers[1] in particular could make immediate use of a wide selection of NLP services, such as summarization, index generation, or question-answering, if they just had access to them from their desktop tools, like email clients, Web browsers, or word processors.

In this paper, we present a strategy for integrating any kind of NLP analysis service into a word processing application, the OpenOffice.org[2] *Writer* program. Based on an existing service-oriented architecture, a plug-in created for *Writer* allows to dynamically find, parametrize, and execute language services. That is, in this work we are not concerned with the development of new NLP services, but rather investigate how *any* existing service can be integrated into an end user's tool. A simplified overview of this idea is shown in Figure 1.

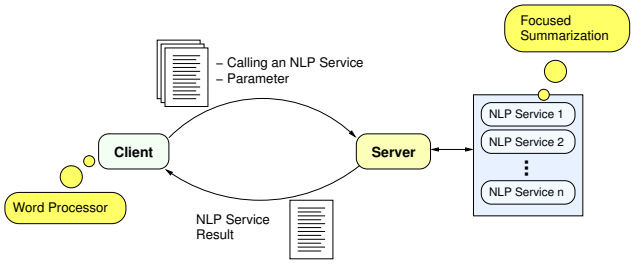

Figure 1: Invoking NLP services directly from a user's client

Our approach allows for a complete reversal of common knowledge acquisition processes: whereas today's knowledge worker has to leave his text processing application to search for relevant information (e.g., through *Google*), process the retrieved results manually, and then continue working on his task, we propose to integrate knowledge retrieval, analysis, and content development into the end-user client. Thus, a user does not have to interrupt his workflow but rather relies on external *Semantic Assistants*, which provide NLP analysis services called directly from the end user's word processing application. These assistants can, for example, search for relevant information on the Web and produce a multi-document summary of the knowledge relevant for the user's current context.

This paper is structured as follows: In the next section,

[1]This term was coined by P. Drucker in 1959. Also known as intellectual worker or brain worker, it emphasizes a worker's capability to work as an expert in a subject matter, rather than, say, through physical labor. See, e.g., http://en.wikipedia.org/wiki/Knowledge_worker.

[2]OpenOffice.org, http://openoffice.org

we describe some application scenarios relevant for our approach in more detail. The design and implementation of our NLP/word processing integration is covered in Section 3. Example applications of the developed solution are presented in Section 4. Related work is discussed in Section 5. Finally, conclusions are presented in Section 6.

2. Application Scenarios

In this section, we describe a number of application scenarios to further motivate our approach of integrating NLP capabilities into a word processor. These scenarios are meant to illustrate the everyday problems that people face in numerous professions and activities. The actors in these scenarios are all knowledge workers, meaning that they have to find, identify, evaluate, and incorporate considerable amounts of existing knowledge in order to do their work.

Scenario 1: Authoring and Analyzing Documents. Editors and journalists continuously face deadlines for delivering articles. Although relevant knowledge—such as previous articles, web pages, and research papers—are readily available online, the typically vast number of hits delivered by an Internet or desktop search makes it impossible to review all documents manually. In this scenario, *focused* multi-document summarization can help the user by presenting an extract of information relevant for the task at hand. The generation of this kind of summary has been investigated for several years within the *Document Understanding Conference* (DUC)[3] competition and lends itself very well to an integration into an office tool: the user can simply highlight a text segment containing questions or other pertinent context information (usually between one and several sentences[4]), which is then used to find relevant documents with an IR system (e.g., from the Web, a digital library, or a local document repository). The summarizer can then prepare a focused multi-document summary of these documents, which contains only those pieces of information relevant for the context question(s). This processing can be performed in the background after the user requested the summary, thereby allowing him to continue work on other parts of his document.
This idea can be further enhanced by adding machine translation tools into the NLP processing pipeline (see the DUC tasks on cross-language summarization), thereby allowing end users access to knowledge in languages they do not speak themselves.

Scenario 2: Information Extraction. Often, knowledge workers require only particular information from a set of documents while working on a task. This might simply be a list of *person* or *company* names or entity relation information, e.g., between *persons* and *events*. This can be achieved with off-the-shelf information extraction (IE) tools, such as the ANNIE system delivered with the GATE framework (Cunningham et al., 2002). Additionally, domain-specific IE can support users in specific knowledge management tasks: For example, a biomedical researcher might need a list of all mutated *proteins* from a set of papers (Witte et al., 2007); and a software engineer might need to find all *method* names covered in a system's documentation (Witte et al., 2008). By integrating IE services, the user can either opt to extract the information from a document he his currently working on, or, like in the previous scenario, on a set of external documents.

Scenario 3: Index Generation. Adding a classical book index to a document is often tedious work, especially when it has not been planned from the start. Simple NLP pipelines can facilitate this task—for example, by performing noun phrase (NP) chunking and building an inverted index from the head noun (first level) and modifier (second level) slots. Using the information extraction techniques mentioned above, specialized indices can additionally be generated, such as person or organization names. Integrated into the word processing application, a draft index can be created directly within the document window, and then further edited and refined by the user.
Index generation can also be applied to *external* documents: instead of looking at text summaries of potentially relevant documents for the current task, a user can also request the creation of a book-type index from documents retrieved through an IR engine, using this as a further navigational aid.

Other NLP Services. The scenarios here are by no means exhaustive—they simply illustrate how enormously useful standard NLP techniques that are already available today can become for an end user when integrated into a standard desktop tool. We expect that future NLP analysis services will be designed directly for deployment in a service-oriented architecture and thereby target the needs of end users even better.

3. Design and Implementation

We now present our solution to the integration of word processing and NLP services. In the first subsection, we briefly describe our service-oriented architecture for NLP/client integration. The second subsection then describes in detail our integration of the OpenOffice.org *Writer* word processor into this architecture. The service-oriented approach was chosen for its ability to model NLP services at a high level of abstraction, thereby hiding the technical aspects of their implementation from the end-user clients. To allow the recommendation of NLP services based on the user's context, i.e., his language capabilities, current task, and client, we provide an ontology model that connects these aspects with available language services.

3.1. Service-Oriented System Architecture

To facilitate the integration of end-user (desktop) clients with natural language processing services, we developed a service-oriented architecture (SOA) that allows to easily connect arbitrary clients with an NLP framework using W3C Web Services.[5] The design and implementation of this architecture is described in (Witte and Gitzinger, 2008). Here, we

[3] Document Understanding Conference, http://duc.nist.gov

[4] An example for such a context is (from DUC 2005): *"What countries are or have been involved in land or water boundary disputes with each other over oil resources or exploration? How have disputes been resolved, or towards what kind of resolution are the countries moving? What other factors affect the disputes?"*

[5] W3C Web Services, http://www.w3.org/TR/ws-arch/

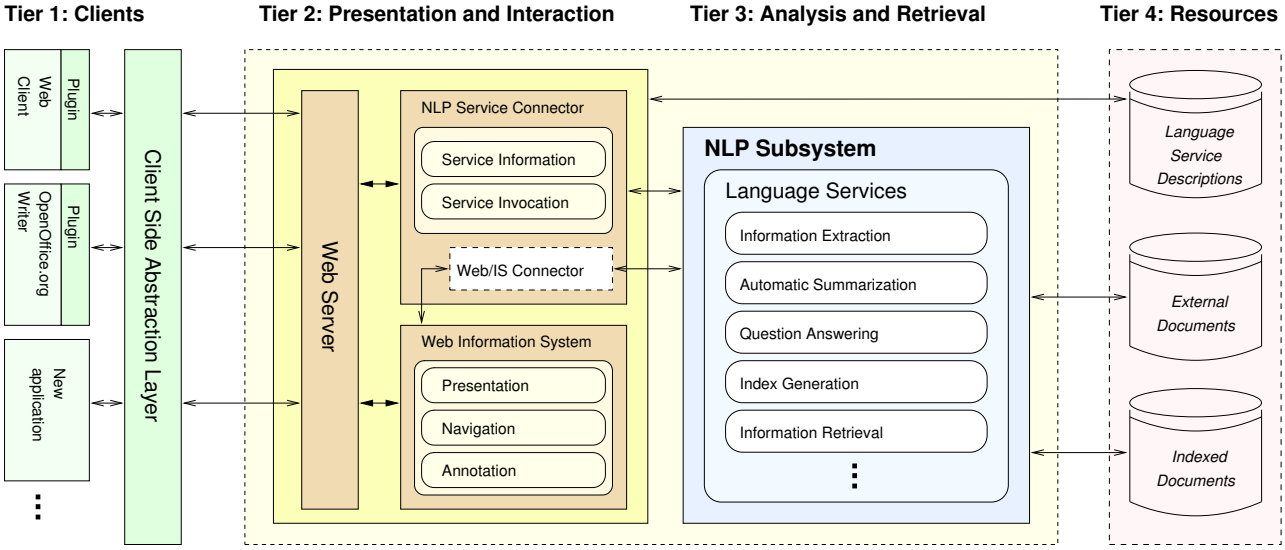

Figure 2: The *Semantic Assistants* architecture for integrating text analysis services and end-user clients

only provide a brief overview to illustrate the steps needed for integrating a new client, in this case, a word processor.
An overview of the architecture is shown in Figure 2. It is based on a typical multi-tier information system design. Clients (Tier 1) provide access to an end-user. Here, we only discuss the integration of a word processor, namely the OpenOffice.org *Writer* application, but our architecture has been designed to allow connections from any kind of client. A client-side abstraction layer (CSAL) provided by our architecture contains a number of pre-defined methods commonly needed for integrating NLP into end-user clients, such as finding available services and handling input/result format conversions. Tier 2 is concerned with handling the interaction between the clients and the NLP frameworks. Since we rely on Web Services for the communication, a web server handles the management of service discovery and invocation, based on WSDL[6] descriptions and SOAP[7] messages. To connect with a concrete NLP framework, our architecture contains an "NLP Service Connector," which is also part of Tier 2. The current version of our architecture supports the GATE framework (Cunningham et al., 2002) for NLP service execution, which forms Tier 3 of our architecture (other frameworks, such as UIMA (Ferrucci and Lally, 2004), can be integrated in the future). Resources form Tier 4, which includes documents stored locally or on a network (including the Internet), as well as metadata about available NLP services, which are formally described using an OWL-DL[8] ontology.
To integrate a new client, the following four steps have to be performed:

1. From the client application, import the Java archive containing our implementation of the client-side abstraction layer (CSAL).

2. If necessary, tell the CSAL the address of the Web service endpoint. The CSAL classes that need to know the address have a default value for it.

3. Create a `SemanticServiceBrokerService` object, which serves as a factory for proxy objects.

4. Create such a proxy object. This is your "remote control" to the Web service. You can call all methods that have been published through the Web service on this object.

We can now discuss how these steps are performed for a concrete client, a word processor, to integrate it into our architecture.

3.2. The OpenOffice.org Writer Plug-in

The OpenOffice.org application suite offers a mechanism to add application extensions, or *plug-ins*. We used this mechanism to integrate OpenOffice.org's word processing application *Writer* with our architecture, and thus equip the *Writer* with Semantic Assistants (Figure 3).
Our primary goal for the *Writer* extension was to be able to perform text analysis on the current document. This text can, for instance, be a large document from which information should be extracted, or a problem statement consisting of a few questions, which serves as input for a question-answering (QA) Semantic Assistant. Especially for the last use case, it must allow a user to highlight part of a document (e.g., a question) and be able to pass only the highlighted part as input to a language service. Furthermore, the extension must offer the possibility to specify parameters that need to be passed to a selected NLP service.
An OpenOffice.org plug-in is basically a zip file with specific contents and certain descriptions of these contents. The files and directories contained in our zip file are shown in Figure 4. Every plug-in has to include a *META-INF* directory, which contains a file called *manifest.xml*. This XML file lists the elements that come with this plug-in; The concrete manifest file for our plug-in is listed in Figure 5. We can see that it defines three *file-entry* elements specifying the type and location of the following files:

[6] WS Description Language, http://www.w3.org/TR/wsdl
[7] Simple Object Access Protocol, http://www.w3.org/TR/soap/
[8] OWL Web Ontology, http://www.w3.org/TR/owl-guide/

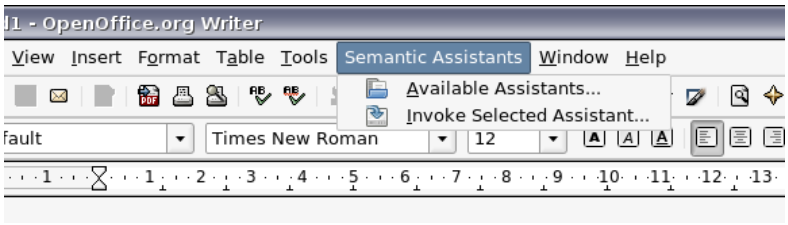

Figure 3: The new "Semantic Assistants" menu entry in OpenOffice.org Writer allows to find and execute NLP services

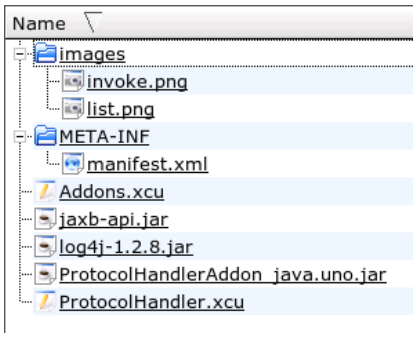

Figure 4: The plug-in file structure

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE manifest:manifest PUBLIC
"-//OpenOffice.org//DTD Manifest 1.0//EN" "Manifest.dtd">
<manifest:manifest
 xmlns:manifest="http://openoffice.org/2001/manifest">
  <manifest:file-entry
    manifest:media-type=
      "application/vnd.sun.star.configuration-data"
    manifest:full-path="Addons.xcu"/>
  <manifest:file-entry
    manifest:media-type=
      "application/vnd.sun.star.configuration-data"
    manifest:full-path="ProtocolHandler.xcu"/>
  <manifest:file-entry
    manifest:media-type=
      "application/vnd.sun.star.uno-component;type=Java"
    manifest:full-path=
      "ProtocolHandlerAddon_java.uno.jar"/>
</manifest:manifest>
```

Figure 5: The *manifest.xml* file for our plug-in

Addons.xcu. This XML file defines how the plug-in should be integrated with OpenOffice.org. In our case, it contains a menu definition, specifying that the menu should only appear in the *Writer* application. For each menu item, we specify which messages should be broadcast throughout the OpenOffice.org runtime system when the menu item is activated.

ProtocolHandler.xcu. This XML file specifies that the messages defined in *Addons.xcu* should be handled by an object of a certain class. This class is provided in the Java archive and must adhere to a certain interface.

ProtocolHandlerAddon_java.uno.jar. This Java archive contains the actual functionality of the plug-in. It holds classes responsible for receiving the messages generated by the menu items, as well as classes responsible for the interaction with the client-side abstraction layer.

Our plug-in creates a new menu entry "Semantic Assistants," as shown in Figure 3. In this menu, the user can inquire about available services, which are selected based on the client (here *Writer*) and the language capabilities of the deployed NLP services (described in service metadata). The dynamically generated list of available services is then presented to the user, together with a brief description, in a separate window, as shown in Figure 6. Note that the integration of a new service does not require any changes on the client side—any new NLP service created and deployed by a language engineer is dynamically discovered through its OWL metadata maintained by the architecture and so becomes immediately available to any connected client.

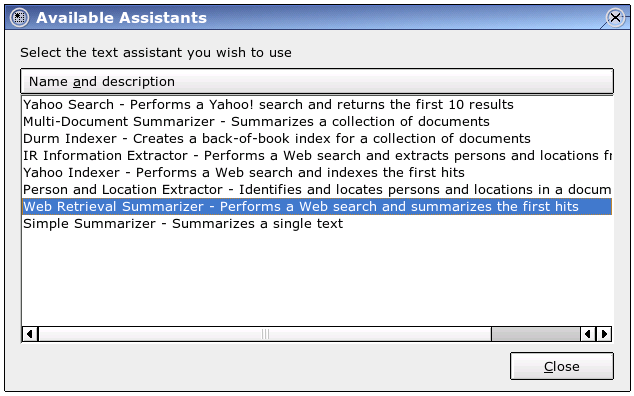

Figure 6: List of available semantic assistants

The user can now select an assistant and execute it. In case the service requires additional parameters, such as the length of a summary to be generated, they are detected by our architecture through the OWL-based service description and requested from the user through an additional dialog window. An example, for the *Web Retrieval Summarizer* assistant, is shown in Figure 7.

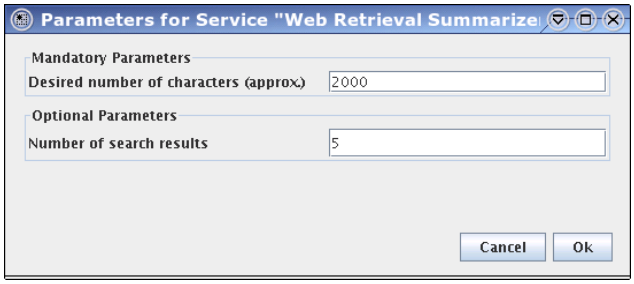

Figure 7: The parameters dialog, which appears when a Semantic Assistant requiring further input is invoked

Once requested, the language service is executed asyn-

chronously by our architecture, allowing the user to continue his work (he can even execute additional services). The sequence diagram in Figure 8 shows the execution of a service through the various tiers described in Section 3.1. Note that all low-level details of handling language services, such as metadata lookup, parametrization, and result handling, are hidden from the client plug-in through our client-side abstraction layer.

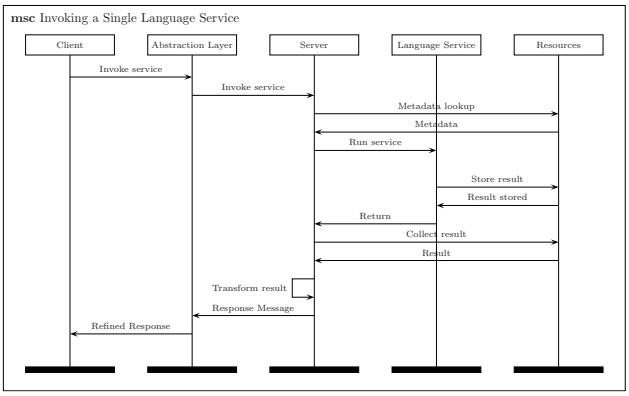

Figure 8: The client invokes a language service and receives a result

4. Application

One direct use case of our Semantic Assistants is to satisfy information needs of a knowledge worker. As motivated in the introduction, language services can deliver focused analysis results directly within the client—here a word processor—needed to perform a task, rather than interrupting the user's workflow by forcing him to perform an external (Web) search.

For example, a user might develop a report on the global climate change and needs information on the role of *"DMSP in the Atlantic marine biology."*[9] With our OpenOffice.org plug-in, the user can simply highlight this phrase in the *Writer* editor window and select the "Web Retrieval Summarizer." This is a compound Semantic Assistant, which performs two functions: In a first step, a selected number of hits from a Yahoo! search using the highlighted phrase is retrieved to build a corpus on-the-fly. This corpus is then fed into the context-sensitive multi-document summarizer ERSS (Witte and Bergler, 2007) to produce a summary. All these actions are performed in the background, allowing the user to continue with other parts of his report. When the summary is ready, the architecture notifies the plug-in, which then presents the generated summary in a new window, as shown in Figure 9. The user can now analyze, edit, or copy parts of the summary within his workflow.

Note that our architecture does not require that the user's word processor and the NLP service reside on the same physical machine. Although this is a possible configuration for personal knowledge management, the underlying Web Service framework also allows to access specific analysis tools from an external service provider. For example, a university might want to deliver question-answering services targeted to its students, answering questions about courses and facilities. A commercial scientific publisher might want to offer a "related work finder" analysis service, similar to the one presented by (Zeni et al., 2007), to scientists writing research papers or proposals.

[9] DMSP stands for "Dimethylsulfoniopropionate" and is a component of the organic sulfur cycle.

5. Related Work

There is not much previous work that deals with the software engineering aspects of integrating NLP services into existing desktop tools.

In their article "Just-in-time information retrieval agents" (JITIR agents), (Rhodes and Maes, 2000) presented a plug-in for the Emacs text editor called the Remembrance Agent (RA). The RA presents, in a special sub-window at the bottom of the Emacs window, a small list of documents that are related to the document currently being written or read. These suggestions can come from multiple different databases or e-mail archives, and are periodically updated. More concretely, every few seconds, an Information Retrieval (IR) process is triggered, performing a search on the specified databases based on, for example, the last 500 words written or the e-mail message that has just been opened for viewing. The results of this IR process are then ranked and listed in the sub-window at the bottom. The second JITIR agent presented in the article, Margin Notes, works by the same principle as the Remembrance Agent. It rewrites displayed Web pages and adds annotations (e.g., links to related e-mail documents) in a separate column on the right to the Web page.

(Colbath and Kubala, 2003) presented TAP-XL, an "automated analyst's assistant." The system's front end is an extension to Microsoft Word. The user writes an initial problem statement, which the system analyzes and uses to retrieve possibly related articles or documents from Internet sources. Unlike with the Remembrance Agent mentioned above, which performs its work in the background without the user actively triggering that work, the user's interaction with the TAP-XL system is more conscious, as he actively poses a problem statement that the system processes. The processing elements that make TAP-XL work form a distributed system of Web Services. Among these services are machine translation, document clustering, multi-document summarization, and fact extraction. The results of these components are stored in a central repository, from where they are accessible to downstream technologies like the word processing front end. The documents to feed this whole system come from a commercial source, as well as from Web harvesting.

These approaches differ from our work in that they are strictly bound to one field of application (e.g., word processing for TAP-XL). By providing an open, client-server, standards-based infrastructure, we can bring NLP to the end user practically regardless of what kind of application she is using. Moreover, the mentioned text assistants' functionality is confined to providing possibly relevant documents. In contrast to that, we want to offer a theoretically open-ended number of NLP services, including machine translation, information extraction, automatic summarization, and automatic indexing. While referenced applications like TAP-XL

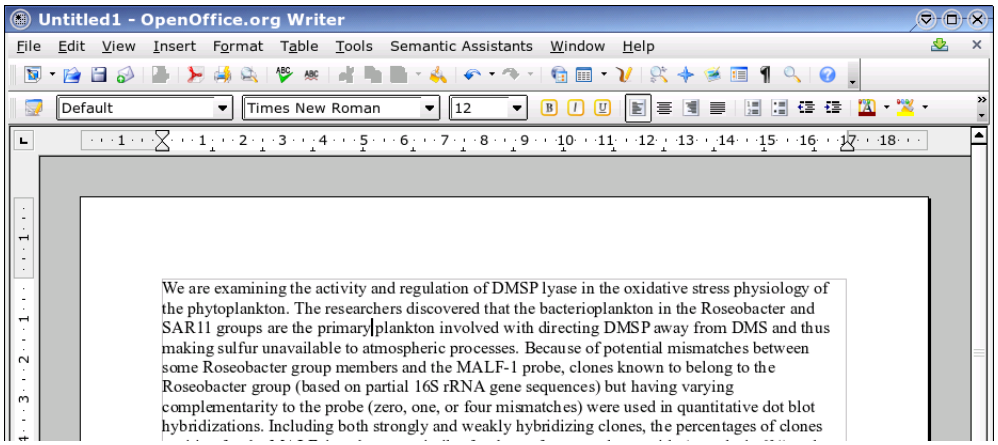

Figure 9: The result of the "Web Retrieval Summarizer" Semantic Assistant, answering the user's question, is displayed as a new document

have their NLP functionality largely built right into them, we follow a different approach in that we clearly distinguish between service requester (the client), the requested service (semantic text assistant), and the underlying architecture. The services offered as part of our architecture are entirely self-contained and unaware of anything that is going on "on the outside." On the other end of the communication, the client program is initially unaware what semantic services it can make use of. It only knows how to talk to the Web Service connecting the two ends. This separation simplifies client development, bears no influence on the NLP development, and allows for higher flexibility.

6. Summary and Conclusions

Today's knowledge workers face constant "information overload." Many NLP techniques and text mining systems have been developed to address the (semi-)automatic analysis of natural language texts—but none of these are of any use to an end user if they are not readily accessible within the applications commonly found on today's desktop environments. In this paper, we described a novel solution for the integration of any kind of NLP service into a word processor application. Our underlying architecture together with the created plug-in perform the "heavy lifting," so to speak, of software engineering work necessary to bring semantic analysis tools to end users. Through its service-oriented approach, it decouples the creation of NLP pipelines and their access from connected clients, so that NLP developers do not need to be concerned with the technical details on how their created services will become available within desktop applications, and client developers likewise do not need to know about the intricacies of developing natural language processing pipelines, since all they see are Web Service descriptions of these services. The developed architecture has been developed based on open standards and open source tools and will also be made available as free software.

Further improvements on the plug-in side will focus on enhanced formatting of NLP results, which can come in diverse formats (e.g., unstructured summaries, structured tables, XML or OWL files). Further plug-ins extending other end-user clients with NLP services are under investigation, like for an email client, Web browser, and software development environment.

On the architectural side, a more detailed handling of the user's current context using our ontological model will allow for a more fine-grained pre-selection and -parametrization of available language services. Together with automated reasoning capabilities of OWL-DL ontology reasoners, an agent-like approach becomes possible, where relevant services are not only executed explicitly, but can also be automatically scheduled by the architecture based on the user's current behavior.

7. References

Sean Colbath and Francis Kubala. 2003. TAP-XL: An Automated Analyst's Assistant. In *NAACL '03: Proc. of the 2003 Conference of the North American Chapter of the ACL*, pages 7–8.

H. Cunningham, D. Maynard, K. Bontcheva, and V. Tablan. 2002. GATE: A Framework and Graphical Development Environment for Robust NLP Tools and Applications. In *Proc. of the 40th Anniversary Meeting of the ACL*. http://gate.ac.uk.

D. Ferrucci and A. Lally. 2004. UIMA: An Architectural Approach to Unstructured Information Processing in the Corporate Research Environment. *Natural Language Engineering*, 10(3-4):327–348.

B. J. Rhodes and P. Maes. 2000. Just-in-time Information Retrieval Agents. *IBM Syst. J.*, 39(3–4):685–704.

Herbert A. Simon. 1971. Designing Organizations for an Information-Rich World. In *Computers, Communication, and the Public Interest*, pages 40–41. The John Hopkins Press.

René Witte and Sabine Bergler. 2007. Fuzzy Clustering for Topic Analysis and Summarization of Document Collections. In *Proc. Canadian A.I. 2007*, LNAI 4509, pages 476–488, Montréal, Québec, Canada, May 28–30. Springer.

René Witte and Thomas Gitzinger. 2008. A General Architecture for Connecting NLP Frameworks and Desktop Clients using Web Services. In *NLDB 2008*, LNCS. Springer.

René Witte, Thomas Kappler, and Christopher J. O. Baker. 2007. Enhanced Semantic Access to the Protein Engineering Literature using Ontologies Populated by Text Mining. *Int. Journal of Bioinformatics Research and Applications (IJBRA)*, 3(3).

R. Witte, Q. Li, Y. Zhang, and J. Rilling. 2008. Text mining and software engineering: an integrated source code and document analysis approach. *IET Software*, 2(1):3–16.

N. Zeni, N. Kiyavitskaya, L. Mich, J. Mylopoulos, and J.R. Cordy. 2007. A Lightweight Approach to Semantic Annotation of Research Papers. In *Proc. NLDB*.

A Personalized Recommender System for Writing in the Internet Age

Mari Carmen Puerta Melguizo*, Olga Muñoz Ramos*, Lou Boves*, Toine Bogers†, Antal van den Bosch†

*Department of Language and Speech, Radboud University.
P.O. Box 9103, 6500 HD Nijmegen, The Netherlands.
M.Puerta, O.MunozRamos, L.Boves@let.ru.nl

†ILK/Language and Information Science, Tilburg University
P.O. Box 90153 NL 5000 LE Tilburg, The Netherlands
A.M.Bogers, Antal.vdnBosch@uvt.nl

Abstract

Writing is a complex task and several computer systems have been developed in order to support writing. Most of these systems, however, are mainly designed with the purpose of supporting the processes of planning, organizing and connecting ideas. In general, these systems help writers to formulate external visual representations of their ideas and connections of the main topics that should be addressed in the paper, sequence of the sections, etc. With the advent of the world wide web, writing and finding information for the written text has become increasingly intertwined. Consequently, it is necessary to develop systems able to support the task of finding relevant information during writing, without interfering with the writing process proper. In this paper we present the Proactive Recommender System: À propos. This system is being developed in order to support writers in the difficult task of finding appropriate relevant information during writing. We raise the question whether the tendency to interleave (re)search and writing implies a need for developing more comprehensive models of the cognitive processes involved in writing scientific and policy papers.

1. Introduction

Writing in a professional environment is a difficult task. Although writing has been practiced for more than 25 centuries, empirical research of the writing process only started some 50 years ago. The first broadly accepted model of the cognitive processes involved in writing was the one proposed by Hayes and Flower developed in the early 80s (Hayes and Flower, 1980). Because text processors were not widely available at that time, it comes as no surprise that they model the cognitive processes involved in writing with pen and paper. Furthermore, since almost all research of the writing process has been conducted in laboratory settings where subjects had to produce short essays, one may ask whether the model can also be applied to writing research papers and policy documents. Finally, the criteria to assess the quality of a short essay are probably very different from those used to assess professional papers. For one thing, writers of professional documents must include or refer to all relevant information that is known, while writers of a short essay are only supposed to cover some items they deem relevant for their exposé. Yet, virtually all software for supporting text production seems to build on the concepts developed in pen-and-paper research.

In this paper we first explain the Hayes and Flower model and its later additions. Then we will relate the model to what is considered good practice for writing in the Internet age, and analyze the ways in which existing writing tools facilitate the tasks and in which ways these tools can be improved. We will illustrate our arguments with a Proactive Recommender System: À Propos. This system is being developed in order to support writers in the difficult task of finding relevant information during writing.

2. The cognitive processes involved in writing

2.1. The Model of Hayes and Flower (1980)

Since the beginning of empirical research scholars have agreed that writing involves at least three different cognitive processes, usually called 'planning', 'translation' and 'review/editing'. Hayes and Flower (1980) propose that there is a recursive interaction between planning, translation and review/editing. The model defines three components: the writing process proper (which includes the three processes/stages mentioned above), the task environment and the writer's long-term memory (see Fig. 1).

Planning involves retrieving domain knowledge from the writer's Long-Term memory (LTM) and organizing it into a plan that specifies, among other things, the effects that the writer wants (or needs) the text to have. During the process of Translating writer's plans and goals are transformed into sentences. In the Reviewing stage the writer evaluates the relation between the text written so far and the linguistic, semantic and pragmatic aspects that best serve the goal.

The task environment includes everything existing outside the writers' mind and that can influence the writing task. The main elements included here are the text produced so far and the so called rhetorical problem (the writing assignment, the specification of topic and the audience).

In the writer's LTM are stored the writer's knowledge about the topic, the knowledge of sources based on literature search, the writing plans and the knowledge about the audience who will read the work.

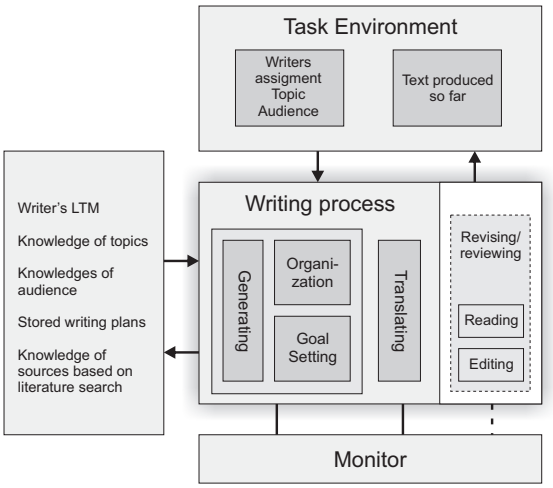

Figure 1: The Model of Writing proposed by Hayes and Flower (1980)

2.2. The Revised Model by Hayes (1996)

Later on, Hayes (1996) revised the model and emphasized the role of working memory, as well as socio-cultural and motivational aspects in writing. The main components are now the task environment and the individual (see Fig. 2). In the new model the task environment is divided into social and physical contexts. The social environment needs to be considered because the way a text is written ought to be affected by the audience it is meant for.

In the physical environment the composing medium or tool used to write is included. Variations in the medium seem to lead to differences in the way people carry out the writing task. For example, Haas (1996) found that writers tend to plan more when they write on paper than with a word processor, presumably because it is easier to sketch, draw and interconnect ideas using pen and paper. Haas also found that writers tend to revise documents on general level (i.e. modifying the structure) when using a pen, and more on a local level (i.e. revision of syntax, semantics, vocabulary) when working on screen. These results suggest that the introduction of computer tools as a medium for writing entice users to change the processes, rather than making the original processes easier or more effective. However, it should be noted that the revised model of Hayes still seems to deal only with writing short essays with pen and paper or with a stand-alone text processor and without the need to look for external information.

The components motivation/affect, cognitive processes, working memory and long-term memory are included in the part that models the writing individual. The main cognitive processes of writing are now text interpretation, reflection and text production. Text interpretation refers to the creation of internal representations based on linguistic and also graphic input. Planning has been replaced with the more general cognitive function of reflection that includes processes of problem solving, decision making and inference. Text production refers of course, to the act itself of producing written texts.

The working memory is also included in the model and in addition to its storage function, it performs a set of control functions. In a similar way, the model of Kellog (1996) integrates working memory into writing. During the writing process information from LTM is retrieved and deposited into working memory.

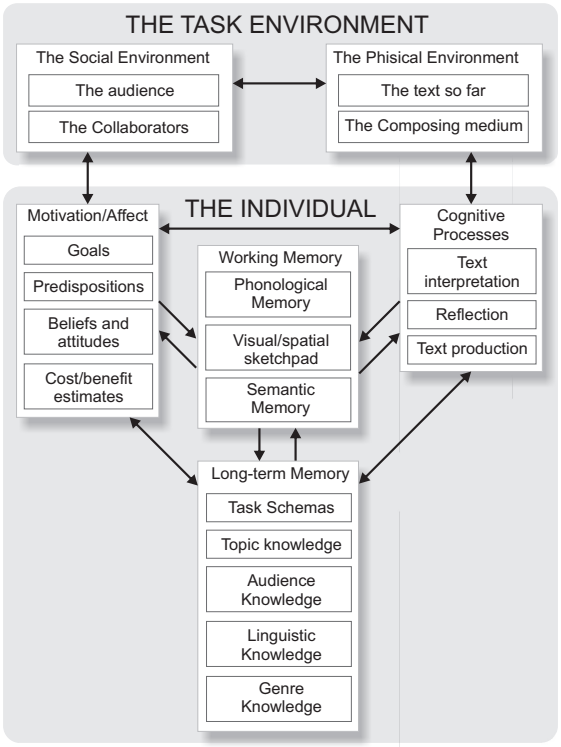

Figure 2: The Model of Writing proposed by Hayes (1996)

2.3. The Role of Long-term Memory and the Need to Look for New Information

Current models of writing assume that knowledge about the topic of the text is available in the writer's neural LTM. The reality of writing professional texts, however shows that writers almost invariably need to look for additional external information. Sometimes it is just very difficult to remember and retrieve information that is already stored in the writer's LTM. The need for searching or verifying detailed factual data is especially important during the reviewing phase. At other times the writer does not know about a specific bit of information and needs to consult external sources.

Of course, the need to do research before writing a paper existed before the advent of digital computers and text processors. Most probably, the idea that Planning should precede Translation derives from that practice. However, the advent of the Internet has caused a dramatic change in the process of writing. More often than not, writing is now interleaved with searching for information. Searching while writing is so easy that few writers can resist it. The question remains, however, whether continuously switching between the tasks of writing and searching is efficient, and whether it tends to result in the best possible quality of the texts. Because it is now common practice to interleave writing and searching, it is not only necessary to design tools

that support the cognitive processes involved in writing, but also tools that help users in finding information that writers cannot retrieve from their own memory.
Most computer writing environments (Britton and Glynn, 1989) seem to have been designed with the purpose of supporting the process of reflection. In general, systems help writers with grammar and spelling or to formulate external representations of their ideas and connect the main issues that should be covered, the ordering of the sections, etc. The Writing Environment (WE) developed by Smith et al. (1987) was one of the first to support the processes of organization and editing by means of a network mode, a tree mode, an editing mode, and a text mode. Another example is the tool developed by Neuwirth and Kaufer (1989), which offers visual representations that help to organize notes and graph summaries.

3. Writing in the internet age: finding related information during writing

The World Wide Web is a rich source of information about virtually any topic and looking for information on websites has become the most popular way to access information. Search engines have become the primary tool for information access in the Internet and company-internal networks. However, searching and navigating is often not as efficient and easy as users might like. Users frequently feel "lost in hyperspace". Feeling lost or the tendency to lose one's sense of location is the most common problem users experience while navigating and strongly affects user's performance and satisfaction (Conklin, 1987; Gwizdka and Spence, 2007; Puerta Melguizo et al., 2006). Ironically, the very potential strength of hypertext, namely improving the management of loose collections of relatively unstructured information, turned out to be a major potential weakness (Neuwirth and Kaufer, 1989).
Furthermore, keyword-based search is inefficient and relevant information may be missed because the writer did not realize that the information exists and could be looked up. Considerable time is spent interacting with low-precision search engines. Consequently, the time in which the author is away from creating the document can affect the total time spent, and the eventual quality of the text. Switching from the text editor to the search engine imposes extra demands on the user's cognitive capacities. A system that can relieve authors from explicit search and switching between applications by means of searching information accurately and recommending this information in a proactive manner would be most welcome.

4. À Propos: a proactive recommender system

The main goal of Proactive Recommendation Systems (PRSs) is to consult large quantities of documents, decide what available information is most relevant for the task, and offer that information without user requests. The decision about what information to offer is mainly based on the text that is currently being written, in combination with personal profiles and profiles of the person's working group. A few PRSs for writing have been developed. For example, the Remembrance Agent (Rhodes, 2000) suggests personal email and documents. Watson (Budzik and Hammond, 1999) performs automatic Web searches based on text being written or read. A problem with these PRSs is that they are developed as search support tools and do not seem to take into account the specific characteristics of the writing task, which can be seriously affected by any type of interruption from the environment.
The goal of the project À Propos is to develop a proactive, adaptive, personalized, just-in-time knowledge management environment for writers in a professional environment. The architecture of À Propos is inspired by other PRSs such as Watson (Budzik and Hammond, 1999) and Stuff I've Seen (Dumais et al., 2003). A detailed description of the system's architecture can be found in (Puerta Melguizo et al., 2007) where the role of the different components of the system such as observers, filters and gatekeepers is explained.
Deshpande found that two main issues need to be addressed if a PRS for writing is to assist rather than distract the users (Deshpande et al., 2006). First, proactive suggestions must be extremely accurate. Second, procedures to identify the different writing stages and related information needs must be created in order to design an appropriate user's interface.

4.1. Selecting and Presenting Relevant Information

Recommendations should be both on topic and personalized. To increase the topicality of suggestions one can use detailed personalized taxonomies integrated in an easily expandable, yet robust IR model (such as the Vector Space model). Ideally, personalization should go so far that two users with different interests writing or reading the same document should get different personalized recommendations. We are investigating two different types of personalization: on the user level and on the group level.

4.1.1. User personalization

From the user perspective we only consider evidence of the user's interests and expertise. From these data we build a personal profile of terms which is used to re-rank the initial recommendations list. Three different sources of information are considered for this purpose. First, the important terms of previously selected documents are added to the user's personal profile. Second, important terms in the user's past documents are given different weights, dependent on whether they were written by the user or merely read. Finally, the PRS allows users to enter informational queries manually; the important query terms entered explicitly are also be included into the user profile.

4.1.2. Group personalization

Group personalization is done on the basis of the expertise of the members of a group. Not every group member has an equal level of expertise or interest in the specific topic of the document being written by an active user. À Propos performs group personalization by identifying the expertise of the group members in different topics. The user-level profile can be seen as an expertise fingerprint of that user, with terms that describe his or her interests. The user's own documents are an effective source for obtaining important

expertise terms (Balog et al., 2007). We can then use taxonomies, such as the ACM hierarchy, to represent the topics for which we want to quantify a group members' expertise. By collecting an adequate number of documents for each topic we can construct topic fingerprints.
The next step is to match these topic fingerprints with the user's expertise profile by calculating the term overlap. A higher overlap indicates more expertise in the subject. This way we can calculate the expertise of each group member on the different topic areas and also find out which group members are experts in the topic of the user's active document. Figure 3 shows an example of such a distribution.
Knowledge of the distribution of expertise over the group is then used for personalization. The recommendation of a document by an expert on the topic should be considered as more reliable and this can have a significant influence on the final re-ranking (Bogers and Van den Bosch, 2006). Group personalization can be used to recommend highly regarded documents that were not in the initial recommendation list. Finally, the expertise fingerprints can also be compared to each other and used to suggest related topics to provide for a more serendipitous experience. Serendipity is especially important in the earliest phases of planning and composing a document.

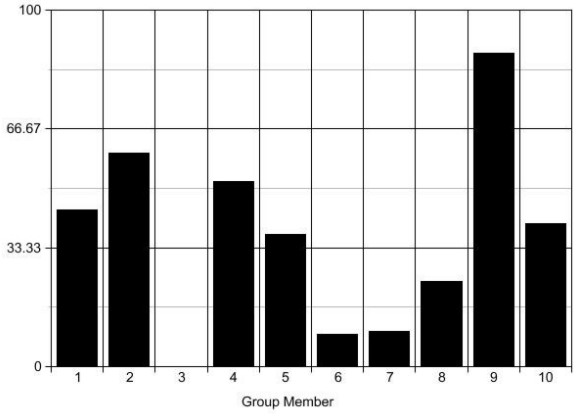

Figure 3: Distribution of expertise in a group. The vertical axis represents overlap between the fingerprints of individual users and experts.

4.2. The Current User's Interface

À Propos proactively submits queries based on the user profile, in combination with what the user is currently typing or reading. The retrieved information is presented proactively and immediately. In the present version of the system, search results are presented in a semi-transparent window located in the bottom right of the screen (see Fig. 4).
The window contains URLs related to what the user is typing. As the user moves the cursor over the references, the URLs become fully visible and active. On clicking the required URL, the user accesses the corresponding paper. The information presented also changes as the user moves the cursor while reviewing previously written parts of the document, on the basis of queries created from the text in the paragraph in which the cursor is located.

4.3. Writing Stages and No Intrusiveness

One problem with presenting proactive information is that it can interrupt the ongoing writing task. Interruptions can be more disturbing and distracting in specific stages of the writing process. This needs to be considered in order for the system to recognize what are the most opportune moments to present the information in a non-intrusive and timely fashion.
Furthermore, replicating other studies (Dansac and Alamargot, 1994; Jones, 2003) it has been found that writers look for extra information especially during Planning and Reviewing (Deshpande et al., 2006; Puerta Melguizo et al., 2007). We studied the effects of presenting proactive information during Planning and Reviewing.

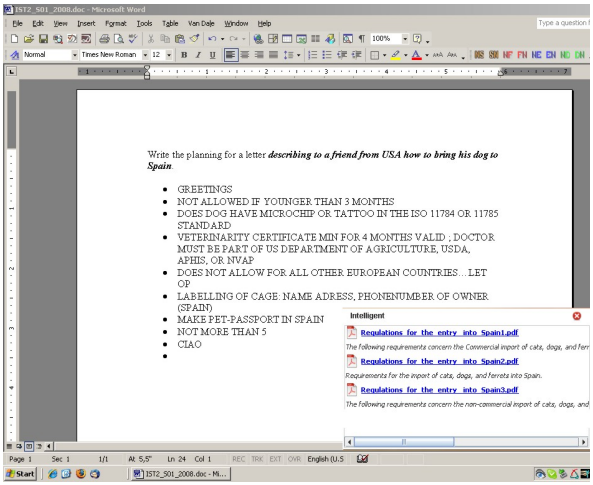

Figure 4: The user's interface.

4.3.1. Presenting Proactive Information during Planning Tasks

Planning involves creating and organizing ideas, and setting goals during composition. In order to simulate the stage of Planning, we used a procedure similar to the one described by Berninger et al. (1996). Participants were told that they had to write short essays about different topics, but before starting to create actual sentences they had to plan and write an outline of the major points of the text with supporting details and the order in which they would be introduced.
A total of 32 participants wrote the planning outlines under four information seeking conditions: 1) without PRS and no option of looking for extra information, 2) without PRS and with the option of getting information by actively searching information in the Web (active search), 3) with presentation of relevant information by our PRS, and 4) with presentation of non-relevant information by our PRS.
We found that the presentation of information by the PRS did not seriously impair time performance during planning. Furthermore, when relevant information was presented proactively, the quality of the resulting writing plan was significantly better, in the sense that participants introduced more information and included more correct ideas in their planning than in any of the other conditions. Concluding, the presentation of proactive relevant information by the PRS improves the quality of the planning process. At

the same time, the interrupts caused by presenting newly retrieved information does not disturb the task, since it does not take more time to write a plan in the condition where the PRS presented relevant information, compared to the other conditions. The results of this experiment also show that active search initiated by the user resulted in a lower quality of the information found and the written text.

4.3.2. Presenting Proactive Information during Reviewing Tasks

When reviewing previously composed text, writers read and edit their written text whenever errors or weaknesses are detected in the text. In this stage writers normally look for factual information in order to correct and/or justify written ideas. To explore the effects of presenting proactive information during the review phase, we asked participants to perform two different editing tasks: spelling corrections and filling in factual information that was left to be specified exactly when the text was drafted. These two tasks are similar to the ones described by Iqbal et al. (2005). Twenty participants performed both reviewing tasks under three different information seeking conditions: 1) without PRS but with the option of getting information by actively searching information in the Web (active search), 2) with presentation of proactive relevant information by our PRS 3) with presentation of non-relevant information by our PRS. Again, the main results of this experiment show that the presentation of proactive information does not seriously impair the time performance in editing tasks in comparison with the conditions in which the user was not interrupted by the system. Furthermore and very importantly, the time spent in looking for new relevant information was shorter when the PRS presented relevant information than in the cases in which participants had to search for the information actively. The information seeking time was even longer when non-relevant information was presented proactively. In this case, after assessing that the information by the PRS could not help in completing the editing task, participants started an active search. This result emphasizes the importance of developing appropriate search profiles and filters as described above. Finally, the quality of the editing tasks was also significantly better when proactive relevant information was presented showing once more, that active search initiated by a user is less effective.

5. Conclusions

Several systems have been developed to support the process of writing. Most of these systems, however, are mainly designed with the purpose of supporting the processes of reviewing and planning, organizing and connecting ideas. We think that it is also necessary to develop systems able to support author's knowledge about the topic. Or in other words, to develop systems that offer an external LTM memory to the writer and that can be accessed whenever the need to enter new information is needed. In this paper we presented the Proactive Recommender System: À propos. This system aims at supporting writers in the task of finding appropriate relevant information during writing.

First, we presented our approach at developing group and personal profiles that make sure the information presented by the system is relevant to the writer and to the specific piece of text is being written. We also describe the studies we performed in order to explore the effects of presenting proactive information when writers are planning and reviewing text. From our experiments we conclude that the user's interface of the PRS does not negatively affect the task of writing. And even more important, when relevant information is presented, the quality of the resulting text significantly improves in comparison with the situations in which the user actively seeks for information.

Futhermore, the results of our experiments with proactive presentation of information suggest that professionals are willing to accept unsolicited pop-up windows and similar interrupts if the information that they are alerted to by those interrupts is relevant for the completion of their (writing) task. Yet, more research is needed to better understand the human factors issues related to these interrupts.

5.1. An External Long-Term Memory

One of our goals is to develop the PRS in such a way that it can be used as an addition to the writer's neural Long-Term Memory (LTM). So far, virtually all writing research has been conducted in settings in which the LTM was limited to the writer's own brain (Berninger et al., 1996; Neuwirth and Kaufer, 1989). The advent of extremely powerful search systems already has had a large effect on the way people consider LTM. Students are no longer trained to memorize facts and information; rather, they are trained in efficient and effective search techniques. Thus, it is becoming more important to know how to find information than to memorize information in the first place. However, access to the virtually unlimited information in the Internet is not without problems. Knowing less, while searching more will make it more difficult to assess the importance of newly found information and to integrate it in a coherent framework. While computer tools may not be able to facilitate the integration process any time soon, they may be able to support the decisions about the relevance of the results returned from a query. The personalization and expert ranking that we are investigation holds the promise that it can help pofessionals to avoid getting lost in hyperspace and cyberspace. Also information retrieved in the form of documents or text snippets may have a different impact on how one decides to organize the information in a coherent text than when the information is retrieved from one's own experience. Consequently, we think it is neccesary to develop a new model of cognitive writing processes in which the external LTM that the WWW and other databases conforms, needs to be included as an important part of the physical environment. In this new model it would be also neccesary to include the need to look for information outside our brain during writing as another important cognitive process to consider. Currently there are a few models that try to represent writing and information searching process. The most relevant is the model of Kuhlthau (2004). In this model however, the writing starts as the search is finished. The real situation however is that both processes interact continuosly while writing and consequently a model that takes this issue into account is neccesary.

6. References

K. Balog, T. Bogers, L. Azzopardi, M. De Rijke, and A. Van den Bosch. 2007. Broad expertise retrieval for sparse data enviroments. In *Proceedings of the 30th Annual International Conference on Research and Development in Information Retrieval (SIGIR, 2007)*, pages 551–558, Amsterdam, The Netherlands. ACM Press.

V. Berninger, D. Whitaker, H.L. Yuen Feng Swanson, and R.D. Abbott. 1996. Assessment of planning, translating, and revising in junior high writers. *Journal of School Psychology*, pages 23–52.

T. Bogers and A. Van den Bosch. 2006. Authoritative reranking of search results. In *Proceedings of the 28th European Conference on Information Retrieval (ECIR 2006)*, volume 3936, pages 519–522.

B.K. Britton and S.M. Glynn. 1989. *Computer Writing Environments: Theory Research and Design*. Lawrence Erlbaum Associates, Hillsdale, New Jersey.

J. Budzik and K. Hammond. 1999. Watson: Anticipating and contextualizing information needs. In *Proceeding of the 62th Annual Meeting of the American Society for Information Science*, pages 727–740.

J. Conklin. 1987. Hypertext: An introduction and survey. In *IEEE Computer Magazine*, volume 20, pages 17–41.

C. Dansac and D. Alamargot, 1994. *Accessing referential information during text composition: when and why?*, pages 76–97. Amsterdam University Press.

A. Deshpande, L. Boves, and M.C. Puerta Melguizo. 2006. À propos: Pro-active personalization for professional document writing. In *SigWriting, 10th International Conference of the EARLI Special Interest Group on writing*.

S. Dumais, E. Cutrell, J. Cadiz, G. Jancke, R. Sarin, and D. C. Robbins. 2003. Stuff ive seen: a system for personal information retrieval and re-use. In *Proceeding of the 26th Annual Int. Conference on Research and Development in Information Retrieval (SIGIR 2003)*, New York. ACM Press.

J. Gwizdka and I. Spence. 2007. Implicit measures of lostness and success in web navigation. *Interacting with Computers*, pages 357–369.

C. Haas. 1996. *Writing Technology Studies on the Materiality of literacy*. Lawrence Erlbaum Associates, Hillsdale New Jersey.

J.R. Hayes and L. S. Flower, 1980. *Cognitive processes in writing*, chapter Identifying the organization of writing processes, pages 3–30. Lawrence Erlbaum Associates, Hillsdale New Jersey.

J.R. Hayes, 1996. *The science of writing: Theories, methods, individual differences and applications*, chapter A new framework for understanding cognition and affect in writing, pages 76–97. Lawrence Erlbaum Associates, Hillsdale New Jersey.

S.T. Iqbal, P. D. Adamczyk, X. S. Zheng, and B. P. Bailey. 2005. Towards an index of opportunity: understanding changes in mental workload during task execution. In *Proceedings of ACM Conference on Human Factors in Computing Systems (ACM CHI 2005)*, pages 311–320.

P.H. Jones. 2003. Distributed information seeking in research collaboration: An extended economy of resources, memory and cognition. In *CSAPC 2003 Workshop*, Amsterdam.

R.T. Kellogg, 1996. *The science of writing*, chapter A model of working memory in writing, pages 57–71. Lawrence Erlbaum Associates, Mahwah, New Jersey.

C.C. Kuhlthau. 2004. *Seeking meaning: a process approach to library and information services*. Libraries Unlimited, Westport CT.

C.M. Neuwirth and D.S. Kaufer. 1989. The role of external representations in the writing process: Implications for the design of hypertext-based writing tools. In *Hypertext 1989*, pages 319–341. ACM Press.

M.C. Puerta Melguizo, V.R. Lemmert, and H. Van Oostendorp, 2006. *Current Research in Information Sciences and Technologies: multidisciplinary approaches to global information systems*, volume 1, chapter Lostness, Mental Models and Performance, pages 256–260.

M.C. Puerta Melguizo, L. Boves, A. Deshpande, and O. Muñoz Ramos. 2007. A proactive recommendation system for writing: Helping without disrupting. In W-P. Brinkman, D-H. Ham, and Wong W., editors, *ECCE 2007: European Conference on Cognitive Ergonomics*, pages 89–95.

B. J. Rhodes. 2000. *Just-in-time Information Retrieval*. Phd thesis, Massachusetts Institute of Technology, Massachusetts, USA.

J.B. Smith, S.F. Weiss, and G.J. Ferguson. 1987. A hypertext writing environment and its cognitive basis. In *Hypertext 1987*, pages 195–214.

Towards Automatic Document Quality Assurance

Neil Newbold and Lee Gillam
University of Surrey
Surrey GU2 7XH, UK
E-mail: n.newbold@surrey.ac.uk, l.gillam@surrey.ac.uk

Abstract

In our work, we are exploring the potential for automation of the quality assurance process as it applies to document production and revision, and how a variety of corpus linguistics techniques, typically used in isolation, can be combined as a means to achieve this. In this paper, we describe developments to date and results obtained, through the implementation and evaluation of a number of software components, including those of the University of Surrey's content analysis applications (System Quirk). These components include a Plain English thesaurus, terminological lookup supported by ISO standards, automatic terminology discovery using statistical and linguistic techniques, and the use of five readability metrics to assess any apparent improvements indicated by these processes. The components were integrated within the existing framework of GATE to demonstrate the potential for wider use of controlled authoring, and analysis was based on standards being produced within the EU eContent project LIRICS.

1. Introduction

A vital aspect of effective communication is ensuring that the right message is being conveyed. Clear and consistent use of language, and especially terminology, will have an impact on the ability of readers to comprehend and assimilate what is written. Over-reliance on ill-defined or author-invented terminology leads to the word "jargon" being applied in a pejorative sense. Restrictions on vocabulary and grammatical structure in controlled authoring systems such as the ASD Simplified Technical English Specification (ASD-STE100) can be vital to the continued maintenance of systems, in this case the upkeep of aircraft, and assume a common and well-governed use of terminology. The restrictions on vocabulary and grammatical structure are closely aligned to the maintenance processes, and the need for critical communications where ambiguity or difficulties in interpreting subclauses could have devastating consequences. A beneficial side-effect in such systems is the improved ease of translation: Boldyreff et al (2001) identify that such improvements may be of benefit also to those with learning difficulties, limited reading skills, dyslexia, and deaf users preferring visual language. In the mainstream, the notion of quality assurance is well established, yet quality assurance per se appears not to extend to the written word. For example, authors of international standards, through ISO, demand that written work be precise and comprehensible. However, there is only a small amount of written "guidance" on how to do so, and where it does exist it is easily ignored. In a wider context, the quality of the written word on the (English) web requires even greater attention.

In our work, we are exploring the potential for automation of the quality assurance process as it applies to document production and revision, and how a variety of corpus linguistics techniques, typically used in isolation, can be combined as a means to achieve this. Such efforts have multiple potential outcomes: (i) improved human understanding and transference of concepts between author and reader; (ii) improved machine processing, including suitability of documents for agents of the semantic web to find, share and integrate more easily; (iii) improved capacity for filtering, using document quality as an additional measure, to avoid opaque texts.

In this paper, we describe the system developed, techniques employed, and results obtained in our work towards automatic document quality assurance. We have implemented and evaluated a number of software components, including those of the University of Surrey Department of Computing's content analysis applications (System Quirk). This includes a Plain English thesaurus, terminological lookup supported by ISO standards, automatic terminology discovery using statistical and linguistic techniques, and the use of five readability metrics to assess any apparent improvements indicated by these processes. These components were integrated within the existing framework of GATE (Cunningham et al, 2002) to demonstrate the potential for controlled authoring, and efforts were based on standards being produced within the EU eContent project LIRICS. These efforts lead us toward the development of an assistive tool, at least for authors of standards but potentially also for authors of other critical communications. Though we have used emerging ISO standards to demonstrate our approach, we do not consider it to be limited only to ISO standards. However, adaptation to other documents requires consideration of, at least, the availability of an existing terminology: provision of such resources alongside the documents may also assist readers in navigating the content.

2. Background

A significant consideration in our work is that of "readability". A variety of measures of readability have been constructed on the basis that determining factors largely comprise sentence length, word length, and in some cases a function of the number of syllables. The results of applying such formulae are computed to provide either the level of education or a difficulty score on a scale

of 1-100. The most common method for readability measurement is the Kincaid Formula, used by the American Navy to judge readability of their technical manuals. The U.S. Department of Defence prefers the Flesch Easy Reading Formula, and most U.S. states require insurance forms to score 40-50 on the test (an "average" English document usually scores 60-70). Another common metric is the Fog Index, which indicates how easy the writing is it is to understand. The Fog Index produces a number denoting the number of years of formal education needed by a person to easily understand the text on the first reading. The "ideal" Fog Index is 7 or 8, with indexes of more than 12 supposedly running a serious risk of not being understood or even read. The so-called Simple Measure of Gobbledygook (SMOG) was named as a tribute to the Fog Index and is widely used, in particular by health authorities such as the Veteran's Association, to assess the educational level needed to fully understand a text. In contrast to these metrics, which all involve the computation of the number of syllables per word, numbers for which can vary, the Automated Readability Index (ARI) uses the number of characters per word and its output also produces an approximation to the U.S. grade level needed to understand the text. The different characteristics of the readability formulae (Kincaid, Flesch, Fog, SMOG and ARI) are presented in Table 1. Further discussion of these formulae and others can be found elsewhere (DuBay 2004; Gillam and Newbold 2007).

	Kincaid	Flesch	Fog	SMOG	ARI
Sentence length	✓	✓	✓	✓	✓
Characters/word					✓
Syllables/word	✓	✓			
Complex words count (more than three syllables)			✓	✓	
Scale	Grade level	0-100	Grade level	Grade level	Grade level
Ideal outcome	7-8 (13-14)	100	7-8	7-8	7-8

Figure 1: Features of the traditional readability metrics

Common amongst these measures is the lack of consideration of prior knowledge as a mitigation for higher scoring, and other considerations of complexity. Furthermore, the formulae measure only the expected reading level or ability of a reader, and take no account of their expertise in the subject matter, or even their intelligence level. Prior knowledge might be indicated by, inter alia, word frequency and existence of a terminology, thesaurus or glossary: commonly encountered words comprising three or more syllables may be more difficult than infrequently encountered words of fewer syllables: consider, e.g. economic vs pithy; a terminology, thesaurus or glossary provided alongside the document may explain apparently complex terms in more familiar ways. Increased complexity could be indicated by longer noun compounds where bracketing (see, e.g. Pustejovsky, Bergler and Anick 1994) could become problematic, and other potential ambiguities may exist, for example with multiple senses in the same domain. Oakland and Lane (2004) identify "reader factors" and "text factors" that contribute to text difficulty, in which background knowledge and lexical knowledge are reader factors, whereas syntactic structure and word rarity are considered under text factors. For these authors, the existence of a terminology, thesaurus or glossary may itself cause a difficulty due to the possible need for multiple inferences in associating terms and definitions, and definitions across terms, contributing to "Cognitive Load". DuBay (2004, p31) identifies a range of prior literature in relation to readability and comprehension involving factors such as idea density and embeddedness. To be able to take proper account of these factors, we first need to be able to assess the document content, and the content of the related language resources. The more immediate goal is to formulate readability metrics that begin to incorporate such considerations.

3. A System for Document Quality

We (re-)engineered a number of System Quirk components for integration with GATE, building on some of GATE's existing processing resources[1]. Components within our pipeline include:

1. Terminology Lookup
2. Linguistic Term Finder
3. Keyword Extractor
4. Statistical Term Finder
5. SimpleText Analyser
6. Annotation Controller
7. Readability Analyser
8. Replacer

The pipeline for these resources is shown in Figure 1. Terminology Lookup (1) makes use of an existing database of terms, interfaced using the Terminological markup framework (ISO 16642). Each term discovered within the document is annotated using this component. These annotations become useful at a later stage. Unknown terms are annotated via two approaches for terminology extraction: linguistically using the Linguistic Term Finder (2), and statistically using the combination of the Keyword Extractor (3) and Statistical Term Finder (4). The Linguistic Term Finder discovers compound nouns in the document according to specified patterns of part of speech annotations (e.g. in Jacquemin 2001). These map to patterns for use with the GATE POS tagger: a compound noun is identified simply as either a noun (or compound noun) preceded by a noun or as a noun (or compound noun) preceded by an adjective. Multiword (noun) expressions which occur in the document with a greater frequency than the selected threshold are annotated. The Keyword Extractor calculates the

[1] The software and associated documents are available at: http://www.cs.surrey.ac.uk/BIMA/Projects/LIRICS/liricsSoftware.html

frequency and weirdness of individual words, as defined by Gillam (2004), using the British National Corpus (BNC) as a reference collection. The so-called "Strathclyde Readability Measure" (Weir and Ritchie 2006) uses the BNC in a slightly different orientation, using frequencies in the BNC as a proxy for the complexity of the text. Our use here incorporates comparison as a means to an end, not the end itself. Our approach compares frequencies in the source text with those in BNC, annotating so-called "weird" words. Annotations can be controlled by the user via thresholds for relative strength of frequency and weirdness. The Statistical Term Finder (4) then uses the "weird" annotations as seeds to examine neighbouring words and identify collocation patterns (Smadja 1993). If a collocating word consistently appears in the user-defined neighbourhood size above a predetermined weirdness threshold, the resulting multiword expression is annotated. This can be made iterative, such that the resulting multiword expression is used to derive longer expressions satisfying the threshold.

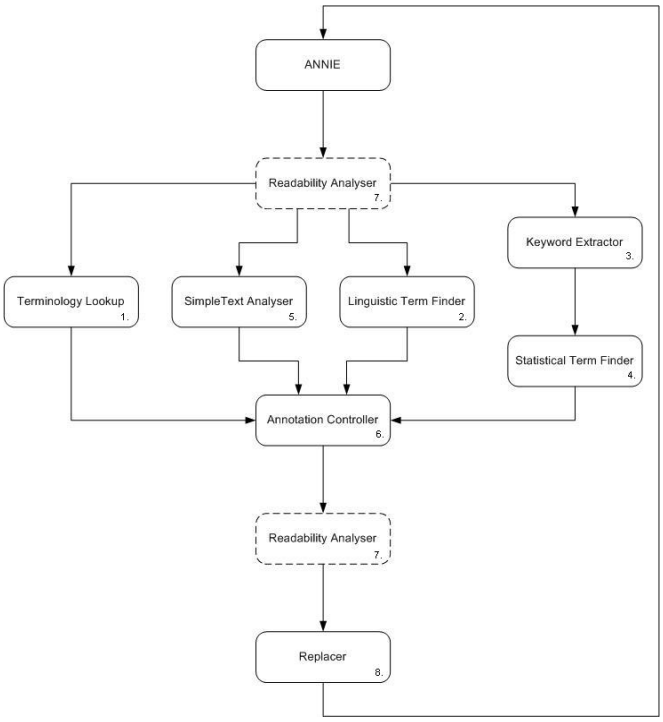

Figure 2: Pipeline

Complex and verbose words and phrases in the document are identified using a Plain English thesaurus, which offers simpler alternatives, through the Simple Text Analyser (5). Alternatives are suggested to the user as annotations, and selected alternatives can be automatically substituted via the Replacer (8). The Annotation Controller (6) collects annotations from the Linguistic Term Finder and the Statistical Term Finder to annotate term candidates with metadata detailing whether the new term originated from linguistic or statistical analysis. Terms which are both linguistically and statistically valid provide a stronger case for addition to the terminology collection. The Annotation Controller also collects annotations from the Simple Text Analyser and optionally allows for annotations to be prioritised such that recognised terms are not additionally annotated as potential new terms, and annotations from the Simple Text Analyser would also be dropped. This means that complex phrases recognised as existing or potential terminology will not be annotated as verbose text by the SimpleText Analyser. The five implemented readability metrics can be run over the input text, and also prior to running the Replacer to determine the likely impact of making replacements. The Replacer only substitutes complex phrases selected by the user. These decisions cannot be automated and require author feedback. The process is iterative, enabling the author to assess the impacts of a variety of modifications.

4.1 Analysis

To demonstrate our analysis, two standards being developed within the LIRICS project, at various stages of the ISO process, have been analysed using this system. The documents are the 'Lexical markup framework (LMF)' (at Draft International Standard stage) and the 'Syntactic Annotation Framework (SynAF)' (at Working Draft stage). Terminology Lookup (1) identified 466 instances of known terms in the LMF document, and 96 in SynAF. By contrast, the combined linguistic and statistical techniques for term identification discovered a further 3712 and 1125 potential instances of undefined terms, respectively. This suggests a particular workload for the author and reader alike. The 'LMF' document was roughly three times the size of 'SynAF', but indicatively has rather more terminological content. The top 10 known terms and their frequencies in 'LMF' and 'SynAF are shown in Table 1.

LMF		**SynAF**	
Term	**Count**	**Term**	**Count**
Class	238	Type	28
Form	99	label	15
Lexical Entry	59	data	14
Word	57	definition	9
Lexicon	46	object	9
Data	44	information	6
Paradigm	39	Merge	2
Paradigm Class	31	Parsing	2
Lemma	30	Read	2
Extension	27	Context	1

Table 1: The top 10 known terms with their frequencies in 'LMF' and 'SynAF'

Candidate terms such as 'syntactic annotation', 'annotation', 'SynAF' and 'morph' were identified as items that may need to be defined, definitions for which would be incorporated within Terminology Lookup and applied at the next iteration. Further filtering of this list is

required, and frequency information can be helpful here also. We may consider whether it is worth having definitions for candidates at lower frequencies, or indeed whether such low frequencies are indicative of other issues with the document. Examples of discovered (bigram) term candidates from LMF are shown in Table 2.

Term	Linguistic Discovery	Statistical Discovery	Count
sense class	Y	N	14
lexicon instance	Y	N	8
core package	Y	N	6
sense instance	Y	N	6
external system	Y	N	5

Table 2: Examples of frequent bigrams in 'LMF'

Discovered terms of increased length at lower frequencies indicate the existence of potentially highly complex expressions. The combination of the two methods of identifying candidate terms allows for readability issues caused by ambiguous bracketing to be highlighted. Such a readability issue can be demonstrated by the first item in Table 3, the "complex knowledge organization system":

1. [complex knowledge] [organization system]: an organization system for complex knowledge, simple knowledge is excluded?
2. [complex][knowledge organization system]: a knowledge organization system that is somehow complicated?
3. [complex knowledge organization] [system]: the system is for an intricately arranged "knowledge organization"?

Term	Linguistic Discovery	Statistical Discovery	Count
complex knowledge organization system	Y	N	4
lmf data category selection procedures	Y	N	2
semantic predicate class section	Y	N	2
dual use mrd metamodel	Y	N	2
dual use mrd package	Y	N	2
multi-layered annotation	N	Y	3
multi-layered annotation strategy	Y	N	2

Table 3: Examples of potential multiword terms that were discovered in 'LMF'

Table 3 also demonstrates term inclusion: "data category" is a term from "ISO 1087-2:2000 Terminology work - Vocabulary - Part 2: Computer applications" and "data category selection" is defined in "ISO 12620:1999 Computer applications in terminology - Data categories". We find 2 instances of "lmf data category selection procedures", which appears to extend this notion somehow. Correct interpretation, however, remains an exercise for the document author. The discovery of the "multi-layered annotation" and its "strategy", or perhaps the "annotation strategy" and its multiple layers, may also suggest the need for the correct interpretation to be made clear.

In analysis using a subset of the Plain English substitutions against a further document, ISO/DIS 12620, we found 183 potential replacements, of which 65 were deemed valid. A sample of these is contained in Table 4. Further analysis needs to be performed on the validity of replacements, if the 'component', 'part' substitution is dismissed; the overall success is increased to over 50%.

Phrase	Replacement	Occurs In Text	Replaced	% Correct
application	use	17	1	5.88%
by means of	by	2	2	100.00%
component	part	68	1	1.47%
comprises	is made up of	4	4	100.00%
consequence	result	1	1	100.00%
essential	important	2	2	100.00%
in conjunction with	with	2	2	100.00%
in order to	to	4	4	100.00%
various	different	10	4	40.00%
within	in	20	14	70.00%

Table 4: Replacements filtered from initial suggestions, with the number of times the replacements were correct throughout the rest of the document

To investigate the extent to which this limited number of substitutions could influence the readability scores of the document, the substitutions were made and the readability analysis was re-run. All readability scores demonstrated slight reductions except for the FOG and SMOG results. The increase in readability scores can be attributed to the fact that some Plain English replacements do not increase readability scores. In fact, the number of words in a document can actually increase due to some of the replacements. The most common example of this occurrence is the substitution of "comprises" for "is made up of". The readability scores before and after the replacements are shown in Table 5.

Score	Before	After
Kincaid	14.753	14.747
Flesch	28.534	28.611
FOG	*17.234*	*17.254*
SMOG	*15.432*	*15.447*
ARI	14.408	14.398

Table 5: Readability scores before and after the SimpleText process

4. Discussion

Draft standards were analysed using the techniques presented, and additional commentary was fed into the

ISO process either in NSB comments, or in interactions with the author/editor of the standard. The analysis of language simplicity and consistency, identification of known and unknown terms, and the generation of "understandability" metrics all demonstrated interesting and potentially highly-valuable results. Human interpretation of, and action upon, the results being produced by these components is still required to varying extents. As such, this is very much a work in progress, and further efforts are needed both to improve the formulation of feedback and to lead to improved automation in the system. Ideally, standards authors would make use of such a system prior to document review in the ISO process, potentially leading to a reduction in the quantity of comments relating to document syntax or terminology.

While typical readability formulae can be useful for comparative measurements, they largely lack consideration of external factors that may make a text more or less easy to understand. In addition, while readability formulae consider short sentences, little by way of other lexical or syntactic features tend to be examined. In our work, we primarily consider terminological complexity and how this might be useful as a means to compute the cognitive load in understanding a new text. Avenues for further research are likely to include: (1) improving SimpleText performance through consideration of local contexts (lexical and/or grammatical); (2) consideration of cognitive load through semantic distance; (3) enhanced statistical (term) detection by improving discrimination of collocation patterns, with related investigations currently being undertaken on the Enron email corpus; (4) production of readability metrics that incorporate the results of the approach outlined, taking account of additional background knowledge that can be provided alongside the document.

During the authoring process in general, the disassociation between author and reader can tend to lead to comments about documents being impenetrable. In the development life cycle of an ISO standard, various iterations of the document are reviewed and commented upon. Versions of the document include the Working draft (WD), Committee draft (CD), Draft International Standard (DIS) and Final DIS (FDIS). Comments are generally provided through National Standards Bodies (NSB), where they are expected to have been validated against principles and methods used for the production of the standards. The amount of hidden effort in the production of standards can be significant: consider the possible commentary from twenty NSBs; the number of people involved with each NSB who read the documents and who provide comments into the NSB, and the potential scale of duplicated comments across NSBs becomes apparent. Comments from NSBs then have to be merged, filtered, and subsequently dealt with by the editor of the standard. This multiple-authoring approach should lead to more understandable documents – however, such documents are mostly authored within the technical domain of the committee, with shared knowledge and understanding having developed within these committees, and distributed across some number of other documents produced previously by these committees. It is likely that a reader would need multiple ISO documents open simultaneously to take some account of such prior work. Increasingly, this is true of most ventures.

5. Acknowledgements

This work has been supported, in part, by the EU eContent project LIRICS (22236) and the UK's EPSRC project REVEAL (GR/S98443/01).

6. References

ASD Simplified Technical English. URL: http://www.asd-ste100.org/. Last accessed 19 Feb. 2008.

Boldyreff, C., Burd, E., Donkin, J. and Marshall, S. (2001). The Case for the Use of Plain English to Increase Web Accessibility. In *Proceedings of the 3rd Intl. Workshop on Web Site Evolution (WSE'01)*.

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

DuBay, W.H. (2004). *The Principles of Readability*, Costa Mesa, CA:Impact Information,: http://www.impact-information.com/impactinfo/readability02.pdf. Last accessed 19 Feb. 2008.

Gillam, L. and Newbold N. (2007) Quality Assessment. Deliverable 1.3 of EU eContent project LIRICS. URL: http://lirics.loria.fr/doc_pub/T1.3Deliverable.final.2.pdf. Last accessed 19 Feb. 2008.

Pustejovsky, J., Bergler, S. and Anick, P. (1994). Lexical Semantic Techniques for Corpus Analysis. *Computational Linguistics* 19(2), pp. 331--358.

Gillam, L. (2004). Systems of concepts and their extraction from text. Unpublished PhD thesis, University of Surrey.

Jacquemin, C. (2001) *Spotting and Discovering Terms through Natural Language Processing.* The MIT Press.

Oakland, T. and Lane, H.B. (2004), Language, Reading, and Readability Formulas: Implications for Developing and Adapting Tests, *International Journal of Testing*, Vol. 4, No. 3, pp. 239--252.

Plain English Campaign. URL: http://www.plainenglish.co.uk/. Last accessed 19 Feb. 2008.

Smadja, F. (1993). Retrieving collocations from text: Xtract. *Computational Linguistics*, 19(1) pp. 143--178.

Weir, G. R. S. and Ritchie, C., 2006. Estimating readability with the Strathclyde readability measure. *ICTATLL Workshop Preprints*, 21-22 August 2006, pp. 25--32.