Editing Medical Data Records with a Natural Language Interface

Richard J D Power*[1] and Christian Pietsch[1]

[1]Department of Computing, Open University, Walton Hall, Milton Keynes MK7 6AA UK

Email: Richard J D Power*- r.power@open.ac.uk; Christian Pietsch - c.pietsch@open.ac.uk;

*Corresponding author

Abstract

Background: Electronic Patient Records (EPRs) contain vast amounts of information in the form of texts, such as letters sent from a hospital to the patient's local doctor (GP). The ability to extract data reliably from these texts would yield advances both in medical research and in individual patient care. In developing effective extraction methods, a crucial step is to create corpora that align samples of the relevant texts with a formal encoding of their meanings. To compile such corpora by human annotation is expensive and time-consuming; the more the task can be automated, the better.

Results: We propose a solution in which the results of an extraction algorithm are corrected using a transparent and fully reliable Natural Language Interface. Information Extraction records are transcoded to a Description Logic A-box and presented through a generated 'feedback text', which a medical expert can edit using the Conceptual Authoring technology.

Conclusions: At present we achieve a weak alignment between the encoded meaning and the original text (i.e., a set of records for the whole text); the next step is to achieve a strong alignment that links each phrase in the original text with the entity in the A-box that encodes its meaning. This technology has a wide range of potential applications, for instance as an editing tool for the Semantic Web.

Background

Through computer technology a massive amount of information is now available in digital form, either on the internet or in private documents and databases. As the digital explosion continues, we become increasingly reliant on computer tools that allow us (so to speak) to pick needles from haystacks, internet search engines being an obvious example. However, remarkable though they are, tools of this kind are limited by their inability to understand the data; essentially, they proceed by comparing strings, with no consideration of what these strings might mean. Herein lies the motivation for the 'Semantic Web' proposal, which seeks to establish standards through which at least some information within a document can be encoded in machine-usable form [1].

There are many examples in the medical domain of information that is coded only as text, not as data, thus reducing its potential usefulness. One important case is the management of information about pharmaceutical products, which was investigated in the PILLS project [2]. For any medicine, a pharmaceutical company must maintain a range of documents presenting information about ingredients, means of delivery, applications, side-effects, and so forth; the documents include instruction leaflets for patients, notes for the formularies consulted by doctors, instructions for pharmacists, and reports for marketing and for regulatory bodies. Where medicines are marketed world-wide, some documents must be translated into over a hundred languages, with regular updates to accommodate product modifications or new research findings. Adding these documents together, we are talking about thousands or perhaps tens of thousands of documents, all based on a comparatively small number of facts. The situation cries out for automation, yet at present all the work has to be performed by human technical authors and translators; automation is not an option *because the data is encoded in text.*

A second example, currently under investigation in the CLEF-SERVICES project [3, 4], concerns the way in which hospitals manage patient records. Hospitals in the UK keep digital archives which include the letters sent by specialists to a patient's doctor. These letters summarise recent developments in an informal style that varies from one specialist to another. Among other things they describe appointments, the results of tests, the treatment given, and the specialist's opinions on diagnosis and future treatment. Most of this information is encoded only in the form of text, and for reasons of privacy has to remain within the private archives of the hospital or local surgery. Considered as a whole, these letters represent a huge information base, potentially invaluable for medical research, yet at present there is no way in which researchers can query it. If the same information were somehow encoded in a machine-usable format, it would be possible for an automatic process to sift out any data that give away the patient's identity, thus avoiding violations of privacy, and to make the remaining data available as a scientific resource.

To achieve these obviously desirable benefits, there seem just two options: either we must provide tools allowing the relevant experts (e.g., doctors) to encode information directly as data, or we must develop reliable software for deriving data from text. In most contexts the former solution is impractical, given the time-pressures under which doctors operate, and the urgent need to communicate clearly with other professionals; consequently the latter approach, automatic Information Extraction (IE), seems the only chance. An IE system takes as input a text, possibly including some formatting mark-up (an HTML source file would be a typical example), and produces as output a set of data records encoding whatever information from the text is deemed relevant. From 1987–1997 IE grew into a major research interest in Language Engineering, under the stimulus of a series of competitions called the Message Understanding Conferences. Each competition focussed on news stories in a specific domain, such as product announcements or management successions; the main task was to fill templates, of a pre-arranged format, describing the reported events. However, even with these simplifications, this proved extremely difficult, with combined precision and recall scores never progressing beyond 50-60% [5]. More recently, the focus in IE has shifted to specific subtasks like named entity recognition and coreference resolution, and researchers have relied increasingly on automatic learning algorithms applied to domain-specific corpora with detailed semantic annotations. At present this seems the most promising direction, but it depends on the availability of large annotated corpora — an expensive undertaking when annotation has to be performed by human experts.

How can we most efficiently build up large semantically annotated corpora for a given domain? The method we are investigating in Clef-Services combines two ideas: first, using an existing IE system [6] to take an initial stab at encoding the meaning of a text; second, correcting and completing the resulting records by a Natural Language Interface, using the technology of Conceptual Authoring [4, 7]. In the context of Clef-Services, the main value of this procedure lies in the creation of a gold-standard corpus for evaluating the IE system and gradually improving it. Other applications might make a more direct use of the formally encoded meaning: for instance, in the Pills project mentioned above, a semantic encoding of a Patient Information Leaflet could serve as the input to a Natural Language Generation system, allowing generation of alternative versions of the text in different styles and languages.

In this paper we focus on the Cake system, now under development as part of Clef-Services. The acronymn describes the combination of two technologies, *Conceptual Authoring* (the focus of this paper) and *Knowledge Extraction* (synonymous with IE) in an efficient tool for semantically annotating a text. In the next section we introduce knowledge editing through Conceptual Authoring, and demonstrate how it has already been applied in a system that allows a domain expert to correct and complete a set of records extracted from a text. We then indicate how in the next phase of the project this system will be extended to allow detailed annotation of the original text. Finally, the wider implications of this approach (e.g., for the Semantic Web) are briefly discussed.

Results

Conceptual Authoring

Conceptual Authoring (also called WYSIWYM[1]) was introduced about a decade ago as a means of creating knowledge bases for Natural Language Generation [7–9]. From the outset it was targetted at knowledge bases represented in Description Logic, with a distinction between T-box (an ontology of concepts and relations) and A-box (a set of assertions about instances of these concepts). The purpose of a WYSIWYM interface is to support a domain expert in building an A-box. Instead of writing formulae in a logical language, or manipulating some kind of graphical interface, the domain expert edits the A-box by progressively building up a text, not by typing in characters, but by choosing from options presented in menus.

Underlying any Conceptual Authoring system is an A-box manager which computes the permitted editing operations given (a) the current state of the A-box, and (b) the constraints imposed by the T-box. The system then generates a 'feedback text' indicating the current state of the A-box, and also generates options for modifying this text, each corresponding to one of the permitted editing operations. These options are presented through menus which open on mouse-sensitive spans of the feedback text. From the user's perspective, the process feels like one of gradually elaborating a text by replacing short phrases with longer ones.

To show how this works, consider the T-box in figure 1, which is a highly simplified version of the ontology actually used in our current program. The notation comes from Erbach's ProFIT system [10]. Subconcepts are introduced by the sign > (thus interventions and investigations are types of event), and attributes by 'intro'; each attribute specification comprises a name (e.g., `has_target`) and a constraint on permissible values (e.g., `locus`). What this means in plain English is that investigations like biopsies and examinations are targetted at a locus — i.e., at some part of the body.

[1]The acronym means 'What You See Is What You Meant'. What You Meant is the content of the knowledge base; What You See is a rendition of this content in natural language, suitable for presentation to a domain expert untrained in the knowledge formalism.

```
domain > [event, object].
event > [intervention, investigation].
intervention > [mastectomy, transfusion]
  intro [has_target : locus, has_indication : condition].
investigation > [biopsy, examination]
  intro [has_target : locus, has_finding : condition].
object > [locus, condition].
locus > [breast, chest, lymphnodes].
condition > [lump, pain, tumour].
```

Figure 1: T-box in ProFIT notation

When starting a new A-box, the user is first invited to choose a root entity, which should be an instance of a suitable abstract concept. In our application this actually embraces a sequence of events, but to simplify the example we will artificially restrict the A-box to a single event. The program's task, accordingly, is to generate some kind of text that describes the current state of the A-box (a slot requiring a filler of type *event*), along with a set of options for adding further content — in this case, by assigning the event a specific type. It can do this by generating a suitable phrase like 'some event', and marking it as a *place-holder* for a specific event yet to be defined. In most applications place-holders have been marked by a colour code, but for convenience we here use square brackets. The place-holder is *selectable* — it lights up in some way when visited by a mouse pointer — and by clicking on it the user obtains a menu of options for specifying the event, expressed through phrases that could potentially replace the place-holder.

S1

FEEDBACK TEXT
[Some event].

OPTIONS
A mastectomy was performed on [some body part] to address [some condition]
A transfusion was performed on [some body part] to address [some condition]
A biopsy was performed on [some body part] and revealed [some condition]
An examination was performed on [some body part] and revealed [some condition]

S2

FEEDBACK TEXT
An examination was performed on *[some body part]* and revealed [some condition].

OPTIONS
the breast
the chest
the lymphnodes

S3

FEEDBACK TEXT

An examination was performed on the breast and revealed *[some condition]*.

OPTIONS

a lump

a pain

a tumour

S4

FEEDBACK TEXT

An examination was performed on the breast and revealed *a lump*.

OPTIONS

Cut

Copy

Step S1 in the above sequence shows the initial state of editing as it would appear to a user. The *feedback text* presents the current state of the A-box, through a phrase ('some event') enclosed in square brackets (to show it is a place-holder) and highlighted in italics (to show it is selected). In the pane beneath, a menu of *editing options* allows the user to introduce an event entity of a specific type — either mastectomy, transfusion, biopsy or examination. The option that the user is about to select is distinguished from the others by italics. From the user's perspective the menu offers a choice of sentence patterns, but in reality it offers a choice of operations on the A-box. When the user selects the option '*An examination was performed on [some body part] and revealed [some condition]*', the program reacts by creating an A-box entity of type `examination` with slots (as yet unfilled) for the attributes `has_locus` and `has_finding`, and regenerating the feedback text, as shown in step S2.

The new feedback text has two place-holders, one for each unfilled slot; if the former is selected by a mouse click, it will be highlighted (italics) and, as before, an appropriate menu of options will be shown for assigning a suitable filler. The options include a phrase for an instance of type `breast`, which the user is about to select. By a series of similar choices (steps S3-S4), the user will soon arrive at a feedback text that fully specifies the event. Note that once an entity has been specified, the corresponding phrase in the text can be selected (e.g., *a lump* in S4) for the operations Cut and Copy, allowing the entity to be removed or pasted into another context.

Usage scenario

Given a database of electronic patient records which contains both structured and (largely) unstructured information, the main goal of CLEF and CLEF-SERVICES is to construct an information resource known as a *chronicle*, which has been defined as '[. . .] a unified, formal and parsimonious representation of how a patient's illness and treatments unfold through time. Its primary goal is efficient querying of aggregated patient data for clinical research, but it also supports summarisation of individual patients and resolution of coreferences amongst clinical documents' [11]. To meet these requirements, the chronicle should include information not usually found in structured medical records, e.g. reasons *why* a treatment was started or stopped; the aim is to recover this kind of information from the unstructured records, such as letters from the hospital to the patient's local doctor.

To evaluate and train an IE system to perform this task, using the currently available techniques, annotated corpora are required to serve as a 'gold standard'. Since annotation requires medical expertise, it must be performed by subject-matter experts who lack the time or inclination to master the formal coding scheme; this is the difficulty Cake aims to overcome, by presenting the content of the annotations through a generated interactive feedback text which the annotator can recognise as a paraphrase of part of the original.

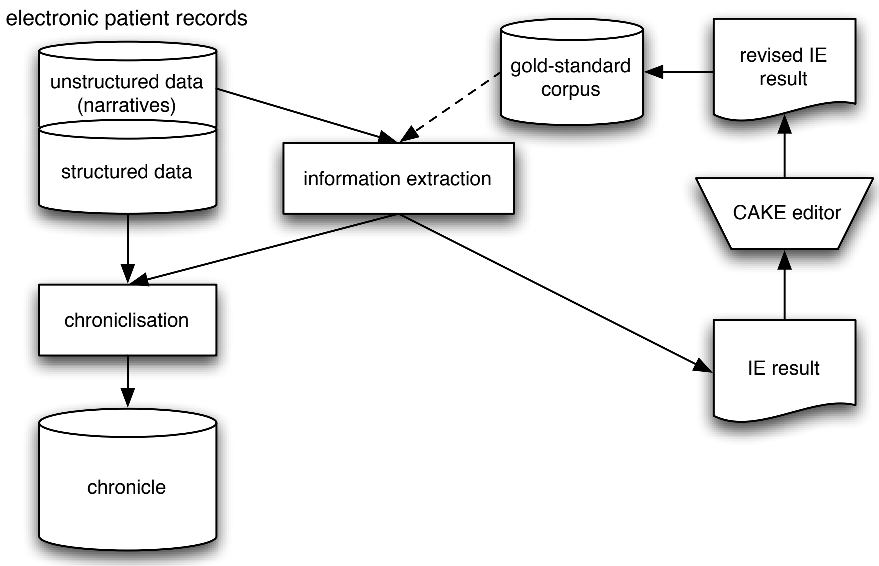


Figure 2: Cake within the Clef-Services data flow.

Figure 2 shows how Cake is connected to the overall data flow in the Clef-Services project: while the structured data can be used directly for chronicle building, the unstructured data first has to be fed into an Information Extraction pipeline (developed by a project partner) [6]. IE results can then be used as an additional data source for building the chronicle. To improve the quality of IE results, we use them as a starting point for creating a gold-standard corpus; the following section will show in detail how this is done.

Demonstration of the prototype

The Cake-1 editor takes two inputs: the text of a human-authored document, such as a letter from the hospital to a local doctor, and an XML file containing records automatically extracted from this letter by the Clef IE component. Figure 3 shows, in the upper pane, the text of an original letter, anonymised so that the actual patient cannot be identified (for example, names and dates are fictitious). The corresponding IE records, some of which are shown below, are presented through a feedback text shown in the lower pane in figure 3; as can be seen from the text, they cover only a fraction of the relevant material in the original letter. The editing task is to make up the missing information; this can be done by clicking on the coloured phrases in the feedback text, which are place-holders for attributes that are currently unspecified (red phrases) or points where further statements can optionally be added (blue phrases).

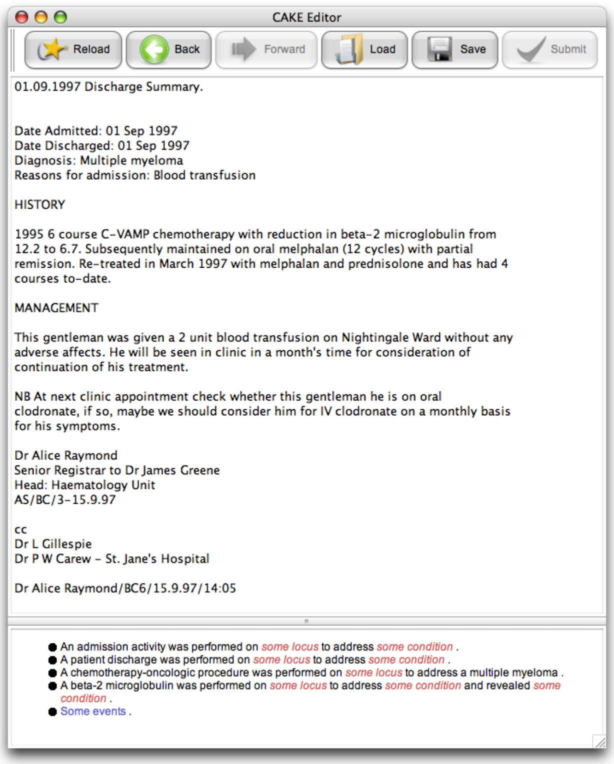
```
<entity id="181" type="Condition" cui="C0026764" />
<entity id="185" type="Drug or device" cui="C0756115" />
<entity id="198" type="Intervention" cui="C0392920" />
<entity id="213" type="Investigation" cui="C0005149" />
<relationship id="221" type="has_indication" arg1Ref="198" arg2Ref="181" />
<relationship id="222" type="has_finding" arg1Ref="213" arg2Ref="209" />
```

In this fictitious example, the IE system has failed to recognise that a blood transfusion was administered. The Cake user can add this event to the records by clicking the place-holder ‘Some events’ (to indicate the intention to add a new event), then clicking on ‘Some event’ (see figure 4) to obtain a list of permitted event types.

If there are many potential fillers, as here, an ellipsis is shown at the bottom of the context menu. A click on it will open a pop-up window with a substring search facility to narrow down the list of fillers. In this case, the user does not need this because ‘blood transfusion’ is listed in the context menu[2]. When the user selects the desired option, the feedback text is re-generated, and now includes a new sentence introducing further place-holders (see figure 5).

Once the user is satisfied that the feedback text reproduces the relevant information from the original letter, the results of editing can be saved in the same XML exchange format that was produced by the IE component.

[2]It is worth noting that the terms displayed for the fillers are just labels that stand for a CUI (concept unique identifier). Internally, entities are only known by their CUI. We use the most preferred term from UMLS [12] to label a CUI in the feedback text and menus; this is not necessarily the term used in the plain-text document.



01.09.1997 Discharge Summary.

Date Admitted: 01 Sep 1997
Date Discharged: 01 Sep 1997
Diagnosis: Multiple myeloma
Reasons for admission: Blood transfusion

HISTORY

1995 6 course C-VAMP chemotherapy with reduction in beta-2 microglobulin from 12.2 to 6.7. Subsequently maintained on oral melphalan (12 cycles) with partial remission. Re-treated in March 1997 with melphalan and prednisolone and has had 4 courses to-date.

MANAGEMENT

This gentleman was given a 2 unit blood transfusion on Nightingale Ward without any adverse affects. He will be seen in clinic in a month's time for consideration of continuation of his treatment.

NB At next clinic appointment check whether this gentleman he is on oral clodronate, if so, maybe we should consider him for IV clodronate on a monthly basis for his symptoms.

Dr Alice Raymond
Senior Registrar to Dr James Greene
Head: Haematology Unit
AS/BC/3-15.9.97

cc
Dr L Gillespie
Dr P W Carew - St. Jane's Hospital

Dr Alice Raymond/BC6/15.9.97/14:05

- An admission activity was performed on *some locus* to address *some condition* .
- A patient discharge was performed on *some locus* to address *some condition* .
- A chemotherapy-oncologic procedure was performed on *some locus* to address a multiple myeloma .
- A beta-2 microglobulin was performed on *some locus* to address *some condition* and revealed *some condition* .
- Some events .

Figure 3: The CAKE editor after importing IE results. *Note: All names and dates in the letter are fictitious.*

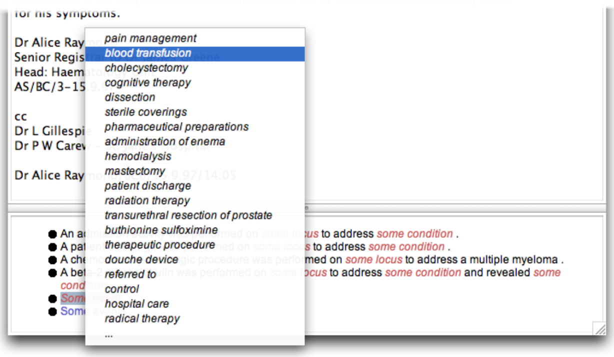
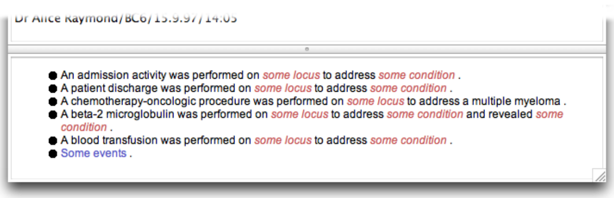


Figure 4: Adding an event in the CAKE editor.


Dr Alice Raymond/BC6/15.9.97/14:05

- An admission activity was performed on *some locus* to address *some condition* .
- A patient discharge was performed on *some locus* to address *some condition* .
- A chemotherapy-oncologic procedure was performed on *some locus* to address a multiple myeloma .
- A beta-2 microglobulin was performed on *some locus* to address *some condition* and revealed *some condition* .
- A blood transfusion was performed on *some locus* to address *some condition* .
- Some events .

Figure 5: New feedback text after adding an event.

Discussion

The purpose of the Cake-1 editor is to yield a correct and complete set of IE records for the original text. By 'correct', we do not mean that the records will capture the entire meaning of the letter, which might contain incidental details beyond the scope of the record templates (e.g., the patient's holiday plans). Nor can we guarantee a successful outcome, since this will depend on the competence of the user as well as the design of the program. What we would claim is that by using this editing interface, a medical expert can produce a correct set of records by constructing a feedback text that paraphrases the essential points in the letter; no knowledge of technical details like template structure is required at all.

We envisage two contexts in which the Cake-1 editor would be used. First, the completed set of records might have high utility, either as a knowledge base for querying, or as a language-neutral semantic representation for NLG. An example of such a context would be the Pills domain mentioned earlier, in which the same information base can support literally thousands of documents, regularly updated, varying in language, purpose, and target reader. Obviously the Clef-Services domain does not fall into this category. Formally encoding the content of the letters has some value, but not enough to justify the huge editing effort. In this second type of context, the value lies in the construction of a corpus for evaluating IE. Successive versions of the IE system can be compared by running them on letters for which the correct answers are known.

Note however that Cake-1 has an important limitation: its outputs can be used for *evaluating* an IE system, but not for *training* it. The reason for this limitation is that Cake-1 delivers only the whole meaning of the whole text, without aligning each part of the meaning with the relevant words or phrases. We can roughly express this difference by saying that text and meaning are aligned only *weakly*; to obtain results useful for training IE, we need an editor that allows a user to specify a detailed or *strong* alignment. Obviously this is a matter of degree. At one extreme (Cake-1), we obtain only a global meaning M for the whole text T, with no mapping of parts to parts. At the opposite extreme, every element in the meaning would be mapped to all its realisations in the text. One could imagine intermediate levels of alignment in which, for instance, records were linked to whole sentences or paragraphs. In the next phase of the project, Cake-2, we aim to extend the editing interface so that it allows users to *transfer* alignment from the feedback text to the original text, thus producing a strongly aligned corpus useful for training as well as evaluation.

Achieving this objective is not just a practical problem: it requires some kind of theoretical model of how meaning and text can be linked. Here we can exploit the formal model of alignment that has been developed for Conceptual Authoring itself. This model applies to what is essentially a Controlled Language — the language generated for feedback texts — so there is no guarantee that it will generalise to texts freely composed by human authors; however, it provides at least a well-defined point of departure.

The purpose of text-meaning alignment in Conceptual Authoring is to allow users to easily select the A-box entities and locations on which editing operations are defined. This is accomplished by consistently adhering to two simple principles:

1. Entities (in a given role) are represented by connected spans of text.

2. Relations between entities are represented by span-subspan relations in the text.

These principles can be illustrated by analysing the feedback text at step S4 of the sequence in the last section, which has three selectable spans:

e_1: An examination was performed on the breast and revealed a lump

e_2: the breast

e_3: a lump

The A-box aligned with this text will have three entities (here notated e_1, e_2, e_3) participating in the following assertions:

$\mathit{type}(e_1, \mathit{examination})$	$\mathit{has\_locus}(e_1, e_2)$
$\mathit{type}(e_2, \mathit{breast})$	$\mathit{has\_finding}(e_1, e_3)$
$\mathit{type}(e_3, \mathit{lump})$	

As can be seen, each entity is represented by a continuous span (which also indicates its type, at least on first mention); the relations `has_locus` and `has_finding` are expressed by the span-subspan relations of the whole sentence (realising e_1) and the constituents 'the breast' and 'the lump'. To put this another way, the sentence frame 'an examination was performed on L and revealed F' tells us that whatever we substitute for L will be the locus of the examination, and whatever we substitute for F will be the finding. In general, if $R(X, Y)$ is a relationship with a domain X and a range Y, then the text span denoting the range Y in the role R is a constituent (subspan) of the span denoting the domain X.

Assuming this model of meaning-text alignment, how can a user transfer alignment from the feedback text to the original? The obvious way of getting started is to select a span in the feedback text, and to highlight spans in the original text that refer to the same entity, using the same mouse-drag operation through which strings are selected for cutting or copying in a text editor. Suppose for instance that the original text runs as follows:

> Mrs Smith's breast was examined. This revealed a lump.

Editing might proceed by selecting the whole sentence denoting e_1 in the feedback text and then separately highlighting 'Mrs Smith's breast was examined' (the first reference to the examination) and the word 'This' (the second reference). A similar process would then link e_2 to 'Mrs Smith's breast' and e_3 to 'a lump'. Obviously there may be difficulties in finding a phrase in the original text with *exactly* the same meaning (e.g., there is no reference to Mrs Smith in the feedback text, since the record templates have no slot for names). Indicating how relations are expressed is also a difficult challenge. The outcome will probably be that we cannot achieve the same level of alignment found in the feedback text; it remains to be seen whether we can achieve annotations that are sufficiently detailed to assist in training an IE system.

Conclusions

We have proposed two methods of obtaining a conceptually aligned text. The first, already implemented in Cake-1, uses Conceptual Authoring in order to construct a new text in a controlled language. The second, to be implemented in Cake-2, assumes that an aligned paraphrase is already available, and allows a user to transfer alignment to a human-authored text by linking phrases that have the same meaning. The first method has been tested over more than a decade and evaluated with expert users [4]; the second is experimental and as yet we have no evidence on whether it will prove effective.

Apart from the specific requirements of Clef-Services, we think this technology is worth pursuing because it will provide a new method for encoding an information base which combines the virtues of data and text — very much as foreseen by the pioneers of the Semantic Web [1]. The advantage of text over data is its transparency to human readers; the advantage of data over text is that it allows services that depend on a formal encoding of meaning, including semantic search, consistency checking, automatic inference, and revision. To elaborate the last of these points, imagine that we have a number of texts describing a product — perhaps a pharmaceutical product as in the Pills domain — all aligned to a common A-box, and that some change in the underlying data is suddenly needed, either because an error is found, or because the information is no longer up-to-date (e.g., research has revealed a new side-effect of the medicine). With aligned texts, it will be possible to make these changes within seconds by automatic revision, rather than laboriously amending what might be hundreds of documents. In the domain of

Electronic Patient Records, the main benefit of alignment would probably be semantic search — the ability to pick relevant passages from lengthy collections of letters and other documents. However, as emphasized above, these benefits often depend on achieving a high level of reliability in IE, and the more immediate objective of the work reported here is to build annotated corpora for training purposes.

Authors contributions

Christian Pietsch wrote the sections on the usage scenario and demonstrator; Richard Power wrote the rest of the paper. Both authors contributed to designing and implementing the prototype.

Acknowledgements

The research was supported by UK Medical Research Council grant G0100852 under the e-Science GRID Initiative. We thank our colleagues in the CLEF-SERVICES project and the clinical collaborators at the Royal Marsden and Royal Free hospitals in London. In particular we acknowledge the help of Angus Roberts and other colleagues at Sheffield University in adapting their XML schema to our requirements.

References

1. Berners-Lee T, Hendler J, Lassila O: **The Semantic Web**. *Scientific American* 2001, **284**(5):34–43.
2. Bouayad-Agha N, Power R, Scott D, Belz A: **PILLS: Multilingual generation of medical information documents with overlapping content**. In *Proceedings of the Third International Conference on Language Resoures and Evaluation (LREC 2002)*, Las Palmas 2002:2111–2114.
3. Rector A, Rogers J, Taweel A, Ingram D, Kalra D, Milan J, Gaizauskas R, Hepple M, Scott D, Power R: **CLEF – Joining up Healthcare with Clinical and Post-Genomic Research**. In *Second UK e-Science "All Hands Meeting"*, Nottingham, UK 2003.
4. Hallett C, Scott D, Power R: **Composing Queries through Conceptual Authoring**. *Computational Linguistics* 2007, **33**:105–133.
5. Gaizauskas R, Wilks Y: **Information Extraction: Beyond Document Retrieval**. *Journal of Documentation* 1998, **54**:70–105.
6. Harkema H, Roberts I, Gaizauskas R, Hepple M: **Information Extraction from Clinical Records**. In *Fourth UK e-Science "All Hands Meeting"*, Nottingham, UK: EPSRC 2005.
7. Power R, Scott D: **Multilingual authoring using feedback texts**. In *Proceedings of the 17th International Conference on Computational Linguistics and 36th Annual Meeting of the Association for Computational Linguistics*, Montreal, Canada 1998:1053–1059.
8. Power R, Scott D, Evans R: **What You See Is What You Meant: direct knowledge editing with natural language feedback**. In *Proceedings of the 13th Biennial European Conference on Artificial Intelligence*, Brighton, UK 1998:675–681.
9. van Deemter K, Power R: **Coreference in knowledge editing**. In *Proceedings of the COLING-ACL workshop on the Computational Treatment of Nominals*, Montreal, Canada 1998:56–60.
10. Erbach G: **ProFIT: Prolog with Features, Inheritance and Templates**. In *Seventh Conference of the European Chapter of the Association for Computational Linguistics*, Dublin 1995:180–187.
11. Rogers J, Puleston C, Rector A: **The CLEF Chronicle. Transforming Patient Records into an E-Science Resource**. In *Fifth UK e-Science "All Hands Meeting"*, Nottingham, UK 2006:630–636.

12. National Library of Medicine: **Unified Medical Language System (UMLS) Metathesaurus, version 2007AB**[http://www.nlm.nih.gov/research/umls/].