Österreichisches Forschungsinstitut für / Austrian Research Institute for / Artificial Intelligence

TR-2008-14

Martin Huber, Jeremy Jancsary, Alexandra Klein, Johannes Matiasek and Harald Trost

Mismatch interpretation by semantics-driven alignment

● Freyung 6/6 ● A-1010 Vienna ● Austria ●
● Phone: +43-1-5336112 ●
● mailto:sec@ofai.at ●
● http://www.ofai.at/ ●

Österreichisches Forschungsinstitut für / Austrian Research Institute for / Artificial Intelligence

TR-2008-14

Martin Huber, Jeremy Jancsary, Alexandra Klein, Johannes Matiasek and Harald Trost

Mismatch interpretation by semantics-driven alignment

The Austrian Research Institute for Artificial Intelligence is supported by the Federal Ministry of Education, Science and Culture.

Citation: Martin Huber, Jeremy Jancsary, Alexandra Klein, Johannes Matiasek and Harald Trost: Mismatch interpretation by semantics-driven alignment. Technical Report, Österreichisches Forschungsinstitut für Artificial Intelligence, Wien, TR-2008-14.

Mismatch interpretation by semantics-driven alignment*

Martin Huber, Jeremy Jancsary, Alexandra Klein, Johannes Matiasek
Austrian Research Institute for Artificial Intelligence†
A-1010 Wien, Freyung 6/6
firstname.lastname@ofai.at

Harald Trost
Department of Medical Cybernetics and Artificial Intelligence
of the Center for Brain Research, Medical University Vienna, Austria
Harald.Trost@meduniwien.ac.at

Abstract

This paper describes a method for the alignment of automatically recognized speech transcripts with formatted documents manually derived from the speech recognition results. Novel features of our alignment method are a parametrizable scoring function, an intelligent tokenization system drawing on domain knowledge, and semantic comparisons. The field of application are dictated medical reports processed by automated speech recognition.

1 Introduction

Current dictation systems aim at a literal transcription of the dictation. Yet even trained persons do not necessarily formulate in the exact form required for the written document because of inherent differences between spoken and written language. This becomes even more problematic for people with less experience. As a result, utterances must be expanded, restructured or reformulated to conform to the required conventions for written reports. Furthermore, recognition errors have to be corrected. Trained typists routinely perform these tasks by editing the draft document.

Future dictation systems will have to move away from simply producing written drafts to producing documents conforming to the formal and informal requirements of texts in their respective domains. Literal transcriptions of the dictations are needed in order to automatically learn recurrent reformulations which turn the written text into the final report. Such transcription corpora are currently scarce as manual transcriptions of large stretches of speech are extremely costly. The goal of the SPARC project is to develop methods for the automatic reconstruction of literal transcriptions from large corpora of pairs of automatically produced draft transcriptions and manually edited final documents. This will be achieved by an in-depth analysis of non-matching stretches of texts which takes into account semantic and phonetic similarity.

A prerequisite for the reconstruction process is an optimal and reliable alignment of document pairs which can be used to identify and categorize mismatches between the two corresponding texts.

The paper is organized as follows: We start by describing the causes for mismatches between the draft and the final document and how these mismatches can be classified. We continue with a description of the various components of our aligner and how they interact, followed by a discussion of preliminary results. Since this paper describes work still in progress, we conclude with an outlook on the next steps to be taken.

2 Categorizing mismatches by source

Systematic mismatches between the result of the speech recognition and a final written document may be the result of three factors involved in different states of the document creation process:

- Mismatches due to the dictation
  In producing a dictation, speakers often hesitate, repeat or correct themselves or give instructions to the system. This leads to a textual form which needs to be translated into a coherent, formatted written document. Knowledge

* This research was carried out in the context of the SPARC project, a joint project by ÖFAI, the Institute for Signal Processing and Speech Communication at the Technical University of Graz, and Philips Speech Recognition Systems. SPARC is funded by the FIT-IT program of the Austrian Federal Ministry for Transport, Innovation, and Technology. More information can be found at http://www.sparc.or.at

† Financial support for ÖFAI is provided by the Austrian Federal Ministry of Education, Science and Culture.

about speech phenomena in general and dictation in particular leads to the identification of this type of mismatches.

- Mismatches due to speech recognition errors
  Particularly in environments where texts have to be dictated under time constraints or in cases where users have little experience with dictating, speech recognition error rates increase. This leads to recognized textual passages which may bear a phonetic but not a semantic resemblance to the dictated texts. These mismatches can only be resolved by using domain knowledge, by interpreting the user's intentions and by consulting information about phonetic similarity.
- Mismatches due to document formatting
  A human transcriptionist, who is familiar with the domain and the requirements concerning the specific type of text, creates the final document by rephrasing and formatting. Knowledge about formatting standards and preferences as well as typical reformulations based on the recognition results can help identify and categorize the mismatches.

Naturally, the mismatches originating from different steps in the document creation process are interlinked: e.g., frequent self-corrections may lead to an error in the speech recognition result which is then corrected and reformulated in the final document. An extended alignment has to take into account these layered sources of mismatches in order to come up with hypotheses concerning the differences (including their origins) between two texts. The shorter two differing passages are, the easier it is to categorize the mismatches and their sources. Simple string matching does not include sophisticated matching criteria. Therefore, the alignment process needs to be extended to better correlate chunks of recognized and written text.

3 Pattern analysis of mismatches

Mismatches resulting from the different stages in the document creation process exhibit particular patterns that can be used to categorize them. E.g., a reformulation may be detected by finding a pair of words (or phrases) that are semantically similar (or equivalent), but phonetically dissimilar. A recognition error might be detected by finding chunks that correspond to each other in the alignment, are phonetically similar, but have unrelated semantics.

A manual comparison was made to identify and classify recurrent patterns of mismatches between the output of an existing automated speech recognition system (ASR) and the final written document. Several typical patterns were found in the process.

One of the most frequent mismatches is based on recognition errors due to *phonetic similarity*: Utterances are replaced by phonetically similar utterances (eg. "are dressed" spoken and "her to rest" recognized). A more complex problem arises in combination with a word that is not contained in the recognition lexicon. Whenever the speech recognition system is confronted with an unknown word, it represents this word with a known word or a combination of several known words that are phonetically similar to the original utterance (eg. "furosemide" spoken and with "for us might" replaced). If no phonetically similar words are found, the recognizer interprets some of the phonemes as hesitations, further complicating the problem.

Reformulations, *expansions* and *completions* are the reasons for most differences between the draft and the final document. The (arbitrary) insertion or deletion of a single word is an often needed process to reformat an utterance into a written text. Nevertheless, some reformulations and synonym replacements are performed only due to transcriptionist's formulation preferences and have no obvious, traceable reason.

Another origin of errors is the need to format texts in order to give them structure. A good example are *headings*. If a heading is spoken in the dictation, the system has to determine which part of the utterance is supposed to be formatted as a heading. If it was not spoken at all and the transcriptionist has added the heading on the basis of formal requirements or due to layout considerations, the mismatch is not caused by a recognition error but rather by subsequent formatting.

Many mismatches are the result of intrinsic *properties of spoken language*, e.g., repetitions, hesitations or interruptions. A related problem are meta-level utterances by the speaker, such as explicit *instructions* on document formatting or document construction. Utterances like "in addition to" or "next number" require some kind of interpretation and execution. These phenomena are hard to deal with because they do not have a direct corresponding representation in the final document.

Lastly, there are mismatches caused by the auto-punctuation component of the recognizer. Commas

and periods are often placed arbitrarily by the automatic system and have to be corrected by the transcriptionist.

4 The Alignment Process

The starting point of our research was a rather simple alignment procedure used for the evaluation of the automated dictation system by determining the word error rate. The correspondence between the recognizer results and the final document is measured by aligning the two texts minimizing the Levenshtein (Levenshtein, 1966) distance.

In Levenshtein distance alignment, relations between two textual units are categorized as

- correct
- substitution
- insertion
- deletion.

Using only string equality as a measure for correctness, we can count the mismatches but we cannot interpret them. Except for the correct case, all other labels correspond to a discrepancy. For our purposes, the nature of the mismatch needs to be indicated as well. Furthermore, alignment between longer stretches of mismatches may become arbitrary with Levenshtein distance alignment because frequent words (e.g. 'a', 'the', 'is') may occur more than once in such a stretch leading to misalignments. Thus, more sophisticated strategies for minimizing and categorizing mismatches in textual passages are needed.

We decided to approach the task by designing a powerful tokenization component which provides both the appropriate granularity as well as a first crude semantic categorization for the textual entities to be matched by the extended alignment algorithm. Tokens are compared regarding their semantic similarity, and phonetic similarity is considered in order to take into account recognition errors [1]. The result of the alignment taking into account semantic and phonetic information results in a more exact and fine-grained alignment and an annotation for the associated patterns of mismatches which were found. This information can then be used as a base for determining the types and sources of mismatches.

[1]Phonetic similarity in the context of the SPARC project, which is not the focus of this paper, is investigated by our project partners at the Technical University of Graz.

In order to allow for extended alignment by introducing and defining constraints, we separated the dynamic programming scheme of the Levenshtein alignment procedure from the scoring function. Thus, in our implementation, the align function takes three arguments, two sequences with elements of arbitrary type, and a scoring function that is to be called with two elements of these two sequences and should return 6 values:

- the penalty of pairing the two elements
- the penalty of pairing the first element with *empty*
- the penalty of pairing *empty* with the second element
- 3 more values holding the labels for the above mentioned 3 cases.

As a simple example, the scoring function for a simple edit distance alignment could be:

```
simplescore(x,y) {
 if ( x eq y )
   then return (0,1,1,"COR","DEL","INS")
   else return (1,1,1,"SUB","DEL","INS")
}
```

However, for the task of error-type aware alignment of recognition results with corrected reports, we use vectors of indices pointing to complex structures holding various features of both the recognized and the written side. Thus, the scoring function can access different features of the items to be aligned (e.g. the phonetic transcription of the item) and, even more important, can also access the context of $item_i$ in question (e.g. by looking at $item_i - 1$). Furthermore, additional measures such as semantic similarity or phonetic similarity can be applied, thus giving considerable room for interpretation and scoring of mismatches. The idea is to give unexplainable mismatches a higher penalty than explainable (or minor) mismatches.

4.1 Tokenization

While reasonably well-performing general purpose tokenizers are meanwhile widely available for common languages such as English, these tokenizers generally fail to identify domain-specific compounds, formatted expressions and formulae as *one* concept or textual unit.

One potential problem with domain-specific tokenization is the effort to adapt it to a new field

of application. Consequently, we have taken much care to implement our tokenizer in such a way as to facilitate its adaption to new domains. Domain-specific tokenization rules are derived from data rather than being hand-crafted. Hence, the framework and methodology created for the tokenization of medical texts can easily be ported to completely different application domains.

Our approach is strongly driven by the availability of two resources that had initially been compiled for other purposes. One is a large morphological lexicon. Since the primary domain in SPARC are medical reports produced in hospitals, the lexicon was extended with a huge amount of medical terminology, most of which was gathered semi-automatically from publicly available medical corpora such as GENIA (Ohta et al., 2002) and UMLS (Lindberg et al., 1993).

Second, from the ASR system we had grammars available that encode the mappings between the written, formatted form of an expression and its various spoken forms. As an example, a blood pressure of '99/75' might be spoken as either of 'ninety-nine slash seventy-five', 'ninety-nine over seventy-five' or 'ninety-nine seventy-five'. Apart from domain-specific concepts such as medical dosages, units and abbreviations the grammar covers also general purpose concepts such as 'date'.

Both resources were compiled into non-deterministic finite-state transducers (FST - see (Mohri, 1997)). The need for an FST instead of a simple finite state automaton arises from the desire to not only define *acceptable* input, but rather transform it into (possibly multiple) annotated strings.

What do these annotations look like? In the case of the morphological lexicon, each given word form transduces to its lemma and the possible Part-Of-Speech tags according to the PENN tag-set (cf. (Marcus et al., 1993)):

```
hurts  ->  hurt+NNS
       ->  hurt+VBZ
```

The deformatting transducer follows the same idea. However, this time, the applicable annotation is chosen from a set of semantic concepts rather than the PENN tag-set:

```
5 mg   ->  five milligram+Q_PHY
```

This tells us that '5 mg' is of type Q_PHY (a physical quantity).

Because of our FST implementation, tokens are not only recognized, but the tokenizer is also able to annotate these tokens. The basic idea of segmenting textual input into tokens using an FST is simple enough:

1. Preparation of the tokenization buffer: Fill the buffer and normalize white-spaces while doing so. The algorithm starts at position 0 of the buffer.
2. At the current position in the buffer, perform prefix search: Enumerate all prefixes at the current position of the tokenization buffer that are contained in the language of the transducer.
3. The longest prefix that matches is the next token. Update the position in the tokenization buffer to point beyond the token just found (if a white-space follows, skip it).
4. repeat steps 2. and 3. until the buffer is empty.

A number of additional issues have to be addressed. The first one is how to use multiple transducers, as is the case within the SPARC project. Our solution is to build two prefix sets in each iteration – one for each transducer – and then choose the longest prefix contained in either of the two sets. This approach is admittedly somewhat arbitrary, however, it reflects the greedy heuristic that was already chosen when selecting the 'best' prefix from *one* set.

Another problem arises from the fact that it cannot be assumed that each single 'word' in the tokenization buffer will be covered by either of the transducers. If the prefix search returns two empty sets, a fallback rule has to be applied which is guaranteed to find a token. For our tokenizer we apply the same heuristics that most general purpose tokenizers apply.

Even if the two prefix sets are not both empty, subtle issues remain. Consider the following contents of the tokenization buffer at the current position:

```
ofloxacin pivaloyloxymethyl ester
^
```

If there is no lexicon entry for either "ofloxacin" or the whole compound, the longest prefix that can be found will probably be "of". Returning the token "of" is definitely not helpful. To avoid this effect, heuristics had to be defined to render specific token

transitions as invalid (the transition between two alphabetical letters being an obvious candidate).

It is planned to migrate the fallback and token transition rules into a third transducer. This way, an additional amount of hard-coded logic can be expressed as mere data and conveniently be edited and adapted to new problem fields.

While greedy tokenization is in general a good heuristic, it may sometimes introduce subtle errors – in particular in combination with the many multiword expressions contained in our lexicon. Consider the lexicon entry "'up to". This entry will result in all occurrences of "up to" in a text being treated as a single token. Usually, this is what we want – but not always:

Correct	Incorrect
Up to 5 mg are indicated .	He gets up to help her .

In the first example, "Up to" takes the role of a single adverb, and so it makes sense to treat it as one token. In the second example, however, "up" is a particle of "gets". It would make more sense to consider "gets up" as one token. But adding "gets up" 5B5Bto the lexicon wouldn't work in all cases, either: consider a sentence like "He gets up to 5 mg."

Admittedly, these examples are carefully constructed to demonstrate the difficulty, and in practice, they do not occur all too often. Still, they show that perfect tokenization is still far from being reached. Some additional heuristics will have to be implemented to catch these cases.

How can tokens be annotated? When performing prefix search, the solutions enumerated by the FST implementation actually include the transductions for each prefix contained in the language of the FST. When the longest prefix is chosen, the transduction suffixes are returned along with it:

Longest Prefix	Transductions	Annotations
infarct	infarct+NN infarct+VB infarct+VBP	NN,VB,VBP
2.	number two+ENM	ENM

Using the approach described above, we achieve good-quality tokenization even for sentences containing highly domain-specific terminology, formattings and punctuation. Compared with general purpose tokenizers, our specialized tokenizer is consistently able to identify coherent concepts and return them as one token:

SPARC Tokenizer	Tokenizer of TreeTagger
On <IN,JJ,NN,RB> December 6, 1999<DAT> blood pressure <NN> was <VBD> 137/77 <BPR> , <,> temperature <NN> 98.8 <NUM_B> , <,> O2 <UNKN> saturation <NN> 95% <Q_PHY> . <.>	On December 6 , 1999 blood pressure was 137/77 temperature 98.8 , O2 saturation 95 % .

Compared to a more fine-grained tokenization as delivered by most general purpose tokenizers, the SPARC tokenization facilitates semantic interpretation of sentences. Since coherent concepts (like 'December 6, 1999' in the example above) are returned as one single token, it is much easier to map the individual tokens of a sentence to a functor-argument structure or semantic grammar rules. In addition, semantic methods are aided by the concept annotations stemming from the deformatting transducer (Q_PHY, DAT, etc.).

The output of the tokenizer serves as the basis for a HMM-based Part-Of-Speech tagger which was adapted to make use of the set of possible annotations for each token.

4.2 Semantic Matching

In order to measure semantic similarity, resources that map words onto some kind of semantic representation are needed. As our application domain is medical reports, specialized medical terminology has to be incorporated into the knowledge sources. The resource we employ for that purpose is the Unified Medical Language System (UMLS, Lindberg et

al. (1993)), that comes with a metathesaurus, a semantic network and a lexicon (SPECIALIST). The morphosyntactic information from the lexicon was worked into the finite-state transducer that is used as a morphological lexicon and also as resource for tokenization (see above).

The metathesaurus is a very large, multi-purpose, and multi-lingual vocabulary database that contains information about biomedical and health related concepts, their various names, and the relationships among them. Unfortunately, the relations between UMLS concepts appear to depend on the particular knowledge source the concept comes from, and the depth it is modeled in that knowledge source. Nevertheless, for checking synonymity of two words or determining a rough degree of their semantic relatedness these relations appear to be sufficient. In addition, all concepts in the metathesaurus are assigned to at least one semantic type from the UMLS semantic network.

Furthermore, for English a high coverage resource, the WordNet lexical database (Fellbaum, 1998), is available. In Wordnet, English nouns, verbs, adjectives and adverbs are organized into synonym sets, each representing one underlying lexical concept; the relations connecting WordNet synsets are quite different from the relations between UMLS concepts. The synsets are linked by different relations, for our purpose the hypernym relation is the most important relation.

A technical problem arises in combining the two resources because of their idiosyncratic and quite incomparable implementation. Therefore we have transferred both resources to a relational database. Thus, despite the different table structure, these resources can be easily accessed simultaneously, and, more important, information from both can be combined in a single query.

As an example, consider the search from synonyms (normalized to lower case) across both resources. Note that the query variables `$qwords` and `$nstrs` are computed from the query word according to the conventions of WordNet and UMLS. The query word is first stemmed, possibly resulting in more than one root forms. In case of multi-word query terms, for WordNet the words are joined with an underscore character, and, for UMLS, a normalized string is constructed (putting each word to lower case and alphabetically sorting them). `$qwords` then contains a list of all query terms formatted according to the WordNet standard, `$nstrs` the list of the UMLS normalized strings. The (slightly simplified) query below then returns the union of all synonyms found in WordNet and the UMLS metathesaurus.

```
select distinct word
  from wn.s
 where synset_id in
       (select synset_id
          from wn.s
         where word in $qwords)
union
select distinct str
  from mrconso
 where cui in
       (select cui
          from mrxns_eng
         where nstr in $nstrs)
   and lat='ENG';
```

The following ordinal scale has been defined in order to obtain a rough measure of semantic similarity of two words:

7 identical (modulo case)

6 same root (only inflection)

5 synonymous

4 derived

3 siblings

2 same UMLS semantic type
or `parent(word1,word2)`
or `parent(word2,word1)`

1 direct hierarchical relation between semantic types

0 no similarity at all

In the above context, `parent(word1, word2)` means that `word1` maps to a concept/synset (inter alia) that is a direct UMLS superconcept or hypernym synset of one of the concepts/synsets `word2` maps to. Two words are siblings, if they share at least one direct UMLS superconcept or hypernym synset. The intuition behind that was to check for measure that is available in both WordNet and UMLS, has a finer granularity than the (rather crude) UMLS semantic type and assures that both concepts have something in common (the "supertype").

The usefulness of that similarity measure and the results obtained so far are discussed in the evaluation section below.

Type	Number	Example
identical (modulo case)	1097	PERINEOPLASTY - perineoplasty
same root (only inflection)	4462	abnormality - abnormal
synonymous	3352	maximum - maximal
derived	338	diabetic - diabetes
formatting	2306	99% - ninety-nine percent
reformulation	4911	enlarged - large

Figure 1: Number of semantic similarity based realignments

5 Preliminary Evaluation

The alignment which incorporates semantic similarity was evaluated on a test corpus consisting of 1747 reports. The error rates in these reports range from 20% to 30%. For individual textual units, the Levenshtein aligned reports contained 1,147,451 alignment tags and with the extended alignments, 1,122,971 alignment tags were assigned. For the 1747 reports, it was possible to classify 16466 alignments and to newly discover 50826 alignments (cf. fig 1).

Although some of the additional alignments may be misleading, the majority represents a good starting point for proposing a relation between two segments of text. The rating of the alignment tags needs to be further optimized.

6 Future Work

While an optimal alignment takes us a large step closer toward our goal of mismatch interpretation, there remain more complex errors and discrepancies between the documents that require a higher-level analysis and some kind of text understanding and interpretation. This is where we will have to employ methods for a semantic matching of complete sentences and chunks of text.

The first method is based on the headings. Titles provide information about the content of the text in the following paragraphs. In the domain of medical reports the number of unique headings is limited. On account of this, recurrent headings allow to make expectations of the following content. For example, under the title “medications” we can expect a list of medications plus a specification of a dose and the form of intake.

A prerequisite for making available this semantic information is the interpretation of the heading itself. Since headings come in many similar but still subtly different phrasings, the first task is to merge the different notations before the classification can be applied. At the moment we follow two approaches: on the one hand using keywords and semantic type information and on the other hand using a grammar. How can we predict the content of a paragraph and use this knowledge to disambiguate the text under a heading: The first idea is to manually compile a special grammar for recurrent contents under a specific topic. Another approach is to automatically learn concepts for the expected content like specific medical terms, lists of particular information, collocations, etc.

All other ideas for semantic matching and text understanding are based on well-established techniques. Yet the fact that there is no annotated trainings corpus eliminates the possibility to use most standard statistical procedures. Thus, our experiments are currently restricted to automatically extracted semantic collocations and semantic role labeling based on phrase structure information.

A second route that has to be explored in improving the alignment and mismatch identification will be the re-segmentation of the recognizer results in some places. While on the corrected report side our specialized tokenization procedure works fine, on the spoken side the recognized words may be totally unrelated to the token structure of the written report, due to recognition errors.

Consider the following two mismatching regions:

1. *KAPPA KDR901* (report)
   cath lab eight the 901 (recognizer)

2. *foramen ovale* (report)
   foraminal valley (recognizer)

While in the first case it is sufficient to concatenate recognized units to obtain phonetically highly similar sequences, thus evidencing the mismatch type of recognition error, the second case is more complicated. Here, the concatenated phonetics have to be split inside the first recognized word if we

want to obtain a phonetically matching sequence: *foramin—al valley*.

However, given the high number of possible recombinations, these reconstruction steps have to be severely constrained both by the phonetic matching and the token structure of the report so that the results can still be handled. Obviously, semantics is not much help in these cases.

7 Conclusion

In this paper, we have described a system which is confronted with a novel task in a very application-oriented setting. By approaching the task of reconstructing an actually spoken dictation from a recognition result and the final document, we have focused on finding an optimal alignment which takes into account semantic and phonetic information. This led the described work away from a simple string-edit-distance alignment towards an extended alignment which includes an annotation regarding the mismatches between the two texts as they were recognized by the system. Identifying and categorizing mismatches requires a layered approach which relies on knowledge about the document creation process, the conventions of the domain, the characteristics of spoken language particularly as used in dictations, and mistakes and ambiguities caused by the speech recognition system. Knowledge about these heterogeneous, complex aspects and their effects on each other is integrated into the alignment process and forms the base for an eventual reconstruction of the spoken utterance.

Although the approach is tailored to our medical application, there are various task-independent features of the alignment process which are potentially important to ongoing work in other applications, i.e. the combination of an analysis of both form and content of documents, the modelling of similarities on the semantic and phonetic level and their relations to different stages of a text in the document creation process.

References

Fellbaum C. (ed.): WordNet: An Electronic Lexical Database, MIT Press, 1998.

Levenshtein V.I.: Binary codes capable of correcting deletions, insertions, and reversals, Doklady Akademii Nauk SSSR, 163(4):845-848, 1965 (Russian). English translation in *Soviet Physics Doklady*, 10(8):707-710, 1966.

Lindberg D.A.B, Humphreys B.L., McCray A.T.: The Unified Medical Language System, *Methods of Information in Medicine*, 32:281-291, 1993.

Marcus M.P., Santorini B., and Marcinkiewicz M.A.: Building a Large Annotated Corpus of English: The Penn Treebank, *Computational Linguistics*, 19:313–330, 1993.

Mohri M.: Finite-State Transducers in Language and Speech Processing, *Computational Linguistics*, 23:2, 1997.

Ohta T., Y. Tateisi, H. Mima, and J. Tsujii: GENIA Corpus: an Annotated Research Abstract Corpus in Molecular Biology Domain, in: *Proc. of the HLT'02*. 2002.