Normalization and Paraphrasing Using Symbolic Methods

Caroline Brun
Xerox Research Centre Europe
6, chemin de Maupertuis
38240 Meylan France
Caroline.Brun@xrce.xerox.com

Caroline Hagège
Xerox Research Centre Europe
6, chemin de Maupertuis
38240 Meylan France
Caroline.Hagege@xrce.xerox.com

Abstract

We describe an ongoing work in information extraction which is seen as a text normalization task. The normalized representation can be used to detect paraphrases in texts. Normalization and paraphrase detection tasks are built on top of a robust analyzer for English and are exclusively achieved using symbolic methods. Both grammar development rules and information extraction rules are expressed within the same formalism and are developed in an integrated way. The experiment we describe in the paper is evaluated and presents encouraging results.

1 Introduction

Work on paraphrase can be seen in two main perspectives: From the analysis point of view, i.e. how to recognize expressions found in texts that convey similar information (we call it normalization), and from the generation point of view, i.e. how to produce a natural language output semantically equivalent to the original phrase.

In this paper, we address the analysis point of view in an experiment we made in the processing of a corpus consisting of a collection of texts from the Agency for Toxic Substances and Disease Registry (ATSDR) describing different toxic products[1]. In these texts, multiple ways of describing toxic products are present (see 2.1 below), which makes this text collection particularly interesting for the task of paraphrase detection. We build a system were documents are processed and give as output a normalized representation of some selected knowledge. The analysis phase can thus be seen as a paraphrase detection phase, as it unifies in a same representation different ways of expressing similar information about toxic products.

We will first describe the corpus on which we work and then the semantic focus of our paraphrase system. The following section is dedicated to the information extraction task which is seen as a paraphrase detection task in the continuity of the task of parsing. Finally we describe the evaluation of the Information Extraction task performed by our system. Future work and improvements are finally discussed.

[1] see http://www.atsdr.cdc.gov.

2 Corpus Analysis and Expected Output

2.1 Corpus study

The corpus on which we work consists of a collection of texts presenting toxic products from ATSDR that are meant to be read by general public. We have concentrated on the first paragraphs containing in average between 6-7 sentences and consisting in the general presentation of a toxic product. They give information about the name, the appearance (colour, smell), some physical properties and possible synonyms of a toxic product. They also explain where the product comes from and for what purposes it is used. Because of the uniformity of the information conveyed in these different texts, the corpus is rich in paraphrases.

For instance, in the text concerning *acetone* we read:
It evaporates easily, is flammable, and dissolves in water.
And in the text concerning *acrolein* we can read:
It dissolves in water very easily and quickly, changes to a vapor when heated. It also burns easily.

Even in the same text, they are some redundancies and a similar idea can be expressed more than once in different ways. For instance, in the text describing *2-Butanone* we can read:
it is also present in the environment from natural sources.
And later:
2-Butanone occurs as a natural product

These few examples illustrates that the kind of texts we work with deal with a restricted semantic domain and contain a large number of reformulations.

2.2 Semantic focus of our paraphrase system

Our goal is to detect and represent some selected information in the corpus presented above. To achieve this, we want to associate a uniform representation with the different wordings of the same information that appears in the texts. We focus on the different ways of expressing the information relative to the appearance, physical properties, synonyms, use and origin of toxic products. Our representation consists of a list of predicates which are detailed below.

- **PHYS_FORM/2**. This predicate is the result of the normalization of strings expressing the physical form of the toxic product. For instance **PHYS_FORM(ammonia,gas)** expresses that the product *ammonia* is a gas.

- **DESCRIPTION_COLOUR/2**. This predicate is the result of the normalization of strings describing the colour of the toxic product. For instance **DESCRIPTION_COLOUR(antimony,silvery-white)** expresses that *antimony* is a silvery-white product.

- **DESCRIPTION_SMELL/2**. This predicate is the result of the normalization of strings describing the smell of toxic product. For instance **DESCRIPTION_SMELL(1.3-butadiene,gasoline-like)** expresses that the product *1.3-butadiene* has a gasoline-like odor.

- **SYNONYM/2**. This predicate expresses that the second argument is a synonym of the first, which is the name of the toxic product. For instance **SYNONYM(acetone,dimethyl ketone)** expresses that *dimethyl ketone* is another name for *acetone*.

- **PROPERTY/5**. The PROPERTY predicate is the result of the normalization of strings expressing physical or chemical properties of the toxic product. For instance, **PROPERTY(acrolein,dissolve,water,in,NONE)** expresses that the product *acrolein* is soluble in water (instantiation of the four first arguments of the predicate), and that we do have precisions about the way this dissolution occurs (last argument *NONE* is not instantiated by a value). For the same product we have **PROPERTY(acrolein,burn,NONE,NONE,easily)** which expresses that the product is flammable and that the localization of the flammability is unspecified.

- **ORIGIN/4** contains the normalized information whether the product is natural or not and where it can be found. For instance, **ORIGIN(ammonia,manufactured,NONE,NONE)** expresses that the product *ammonia* is man-made, and **ORIGIN(amonnia,natural,soil,in)** expressed that the same product can also be found naturally in soil.

- **USE/6** is the result of the normalization of the uses of the described product. In this first stage we only concentrate in uses where the product is used alone[2]. For instance **USE(benzidine,NONE,NONE,produce,dye,past)** expresses that in the past (last argument is *past*) the product *benzidine* was used to produce dyes (4th and 5th arguments) while **USE(ammonia,smelling_salts,in,NONE,NONE,present)** expresses that *ammonia* is now (last argument is *present*) used in smelling salts (the purpose of the use is not specified here).

[2] In the texts, uses of a product when it is mixed with another can also be described but we decided to ignore this information.

To each of the above-mentioned predicates a suffix _**NEG** can be added if there is a negation.

3 Paraphrase detection

Paraphrasing means to be able, from some input text that convey a certain meaning, to express the same meaning in a different way. This subject has recently been receiving an increasing interest. For instance, Takahashi et. al. (Takahashi et al., 2000) developed a lexico-structural paraphrasing system. Kaji et al. developed a system which is able to produce verbal paraphrase using dictionary definitions (Kaji et al., 2000) and Barzilay and McKeown showed how, using parallel corpora of English literary translations, they extract paraphrases (Barzilay and McKeown, 2001). Paraphrase detection is a useful step in many NLP applications. For instance, in multi-document summarization, paraphrase detection helps to identify similar text segments in order that the summary become more concise (McKeown et al., 1999). Paraphrase detection can also be used to augment recall in different IE systems.

In our experiment, paraphrase detection is a step in normalization, as we want to instantiate the same way the predicates presented above when the informative content is similar. For instance, we want to obtain the same normalized predicate for the two utterances *ProductX is a colorless, nonflammable liquid* and *ProductX is a liquid that has no colour and that does not burn easily* namely:

DESCRIPTION_COLOUR(ProductX,colorless)
PHYS_FORM(ProductX,liquid)
PROPERTY_NEG(ProductX,burn,NONE,NONE,NONE).

The input to our paraphrase detection system is the whole paragraph that describes the toxic product. The analysis of the paragraph produces as output the set of normalized predicates. This output can be produced either in simple text format or in an XML format that can feed directly some database.

The paraphrase detection system is based on three different modules that are described in the following subsections. As claimed in (Takahashi et al., 2000) and for the purpose of re-usability, we distinguish what is of general linguistic interest in the paraphrasing task from what is clearly domain dependent, so these three modules are:

- A general English dependency parser;
- A general morpho-syntactic normalizer;
- A specific- and application-oriented normalizer.

3.1 General English dependency parser

This component is a robust parser for English (XIP) (Aït-Mokhtar et al., 2002) that extract syntactic functionally labeled dependencies between lexical nodes in the text.

Parsing includes tokenization, morpho-syntactic analysis, tagging which is performed via a combination of hand-written rules and HMM, chunking and finally, extraction of dependencies between lexical nodes.

Dependencies are binary relations linking two lexical nodes of a sentence. They are established through what we call deduction rules.

Deduction rules

Deduction rules apply on a chunk tree and consist in three parts:

- Context
- Condition
- Extraction

Context is a regular expression on chunk tree nodes that has to be matched with the rule to apply.

Condition is a boolean condition on dependencies, on linear order between nodes of the chunk tree, or on a comparison of features associated with nodes.

Extraction corresponds to a list of dependencies if the contextual description and the conditions are verified.

For instance, the following rule establishes a SUBJ dependency between the head of a nominal chunk and a finite verb:

```
| SN{?*,#1[last:+]},
  ?*[verb:~],
  SV{?*, #2[last:+]}|
if (~SUBJ(#2,#1))
SUBJ(#2,#1).
```

The first three lines of the rule corresponds to context and describe a nominal chunk in which the last element is marked with the variable #1, followed by anything but a verb, followed by a verbal chunk in which the last element is marked with the variable #2. The fourth line (negative condition: $\sim$) verifies if a SUBJ dependency exists between the lexical nodes corresponding to the variable #2 (the verb) and #1 (the head of the nominal chunk). The test is true if the SUBJ dependency does not exist. If both context and condition are verified, then a dependency SUBJ is created between the verb and the noun (last line).

An important feature is that our parser always provides a unique analysis (determinism), this analysis being potentially underspecified.

3.2 General morpho-syntactic normalization

The morpho-syntactic normalizer is a general module that is neither corpus- nor application-dedicated. It consists of hand-made rules that apply to the syntactic representation produced by our parser. It uses well known syntactic equivalences such as passive-active transformation and verb alternations proposed in Levin. It also exploits the classification given by the COMLEX lexicon (Grishman et al., 1994) in order to calculate the deep-subject of infinitive verbs.

For instance the utterance *Antimony ores are mixed with other metals* is finally represented with a set of normalized syntactic relations expressing that the normalized subject (SUBJ-N) of the verb mix is unknown, and that *mix* has two second actants (OBJ-N) *ore* and *metal* :

SUBJ-N(mix,SOMEONE)
OBJ-N(mix,ore)
OBJ-N(mix,metal)

For this example, both passive transformation and reciprocal alternation transformation have been applied on the set of dependencies produced by the general parser.

Deep syntactic rules are expressed using the same formalism than general syntactic rules presented in the previous section. For instance the following rule construct an OBJ-N (Normalized object) dependency between the surface syntactic subject and a verb in a passive form[3].

```
if ( SUBJ(#1,#2)
     & VDOMAIN[passive](#1,#3)
)
OBJ-N(#3,#2)
```

Unlike Rosé's approach (Rosé, 2000) which also developed a deep syntactic analyzer, this is done exclusively by hand-made rules based on the previous calculated dependencies on the one hand and syntactic and morphological properties of the nodes involved in the dependencies on the other hand.

Together with the exploration of syntactic properties, we also take advantage of morphological properties in order enrich our deep syntactic analysis. This is done using the CELEX database (Celex Database, 2000) by pairing nouns and verbs that belong to the same morphological family, which allows us to obtain for the expression *John's creation of the painting*, the same deep syntactic representation as for *John creates the painting*.

As a result of the second stage, we obtain new deep syntactic relations, together with the superficial syntactic relations calculated by the general parser:

- SUBJ-N (Normalized subject) that links the first actant of a verb (finite or non-finite) or of a predicative noun to this verb or noun.
- OBJ-N (Normalized object) that links the second actant of a verb (finite or non-finite) or of a predicative noun to this verb or noun.
- ATTRIB (General attribute) that links two nodes when the second one denotes a property of the first one.
- PURPOSE that links a verb to its actant expressing the purpose of the action.

It is important to note that predicative nouns are represented by their underlying verbs. e.g. The invention of the process is represented by OBJ-N(invent,process).

[3]VDOMAIN links the first element of a verbal chain to the last element of a verbal chain and passive is a feature that is added to this relation.

3.3 Application and corpus specific normalization

Application and corpus specific normalization is a follow-up of the previous module. But while general normalization is purely based on syntactic transformations and some derivational morphology properties, synonymy relations and all further possibilities of morphological derivations are not exploited. This extension uses the results obtained at the previous analysis level.

The application- and corpus-oriented analysis is organized in two axes that are detailed below.

- corpus oriented linguistic processing;
- corpus oriented paraphrasing rules.

3.3.1 Corpus oriented linguistic processing

We exploit the corpus specific properties at different stages of the processing chain in order to improve the results of the general syntactic analysis. Below are the additions we made:

- Specific tokenization rules.

Since toxic products can have names like *2,3-Benzofuran*, which the general tokenizer does not consider as one unique token, we add a local grammar layer dedicated to the detection of these kinds of names. In other words, this layer composes together tokens that have been separated by the general tokenizer.

- Specific disambiguation rules valid for this kind of corpus but not necessarily valid for all kinds of texts.

For instance, the word *sharp* has *a priori* two possible part-of-speech analyzes, noun and adjective, and we want to keep these two analyzes for the general parser. But, since the noun *sharp* belongs to a certain domain (music) that has no intersection with the domain handled by the corpus, we add specific disambiguation rules to remove the noun analysis for this word.

- Improved treatment of coordination for this kind of text.

The corpus contains long chains of coordinated elements and especially coordination in which the last coordinated element is preceded by both a comma and the coordinator. Since some elements have been typed semantically, we can be more precise in the coordination treatment exploiting this semantic information.

- Adding some lexical semantics information

For the purpose of the application, we have semantically typed some lexical entries that are useful for paraphrase detection. For instance, colour names have the features $colour : +$ added.

- Automatic contextual typing

Some of the manually semantic typing (previous point) allows us to indirectly type new lexical units. For instance, as formulations like *synonyms*, *call*, *name*, *designate* are marked as possible synonymy introducers, we are able to infer that complements of these lexical units are synonyms. In a similar way, syntactic modifiers of lexical units that have been marked in the application lexicon like *smell* and *odor* are odor descriptions. In these cases, direct typing cannot be achieved. For example, the huge number of potential smellings (*almond-like, unpleasant*, etc.) cannot be code by hand. However, the inference mechanism enable us to extract the required information.

- Ad-hoc anaphora resolution.

In our corpus, the pronoun *it* and the possessive *its* always refer to the toxic product that is described in the text. As we do not have any anaphora resolution device integrated to our parser, we take advantage of this specificity to resolve anaphora for *it* and *its*.

3.3.2 Corpus oriented paraphrases

Paraphrases are detected by hand-made rules using lexical and structural information.

Lexical relations for paraphrasing

As mentioned before, in our general normalizer some nouns and verbs belonging to the same morphological family are related. We extend these relations to other classes of words that appear in the corpus. For instance, we want to link the adjective *flammable* and the verb *burn*, and we want the same kind of relation between the adjectives *soluble*, *volatile*, *mixable* and the verbs *dissolve*, *evaporate* and *mix* respectively. We declaratively create a relation (**ISAJ** relation) between these pairs of words, and this relation can then be handled by our parser exactly like a dependency relation which has been previously calculated. Other lexical relations between synonyms (e.g. *call* and *name*) or non-related morphological nouns and verbs (as for instance the noun *flammability* and *burn*) are created.

The lexical relations we created are the following

- **ISAJ** links an adjective and a verb when the verb can be paraphrased by *BE+adjective*
- **TURNTO** links a noun and a verb when the verb can be paraphrased by *TURN TO+noun*
- **HASN** links a noun and a verb when the verb can be paraphrased by *HAVE+noun*
- **SYNO** links two words belonging to the same morpho-syntactic class when the first is a synonym of the second[4].

Normalization rules

Once these relations are created, we can then exploit them in rules.

For instance, the following rule[5] (see below) allows for the creation of the predicate **PROPERTY(aniline,dissolve,NONE,NONE,NONE)** for the utterance *aniline is soluble*.

```
if (
  SUBSTANCE(#1) &
  ATTRIB(#1,#8[adj_property]) &
  ISAJ(#9,#10) &
  #8[lemme]:#9[lemme]
)
PROPERTY(#1,#10,##Pron[lemme=NONE],
##Pron[lemme=NONE],
##Pron[lemme=NONE]
)
```

The rule formalism is the one used for the general syntactic grammar and the deep syntax grammar. In this case, we only have two parts in the rule (*Condition* and *Extraction*, *Context* being omitted). In the present example, since we have detected that *aniline* is the described toxic product (**SUBSTANCE(aniline)**), since an **ISAJ** relation exists between *soluble* and *dissolve* (**ISAJ(soluble,dissolve)**) and finally since the deep syntactic analysis of the sentence has given to us the dependency **ATTRIB(aniline,soluble)**, the final predicate is created.

[4]Since we work in a very specific domain, we have no problem of word-sense ambiguity here.

[5]Variables in a rule are represented by #n.

3.4 Example of output

When applied on an input text describing a toxic substance, such as the following one :

Acetone is a manufactured chemical that is also found naturally in the environment. It is a colorless liquid with a distinct smell and taste. It evaporates easily, is flammable, and dissolves in water. It is also called dimethyl ketone, 2-propanone, and beta-ketopropane. Acetone is used to make plastic, fibers, drugs, and other chemicals. It is also used to dissolve other substances. It occurs naturally in plants, trees, volcanic gases, forest fires, and as a product of the breakdown of body fat. It is present in vehicle exhaust, tobacco smoke, and landfill sites. Industrial processes contribute more acetone to the environment than natural processes.

the system is able to extract the following list of predicates:

```
SUBSTANCE(acetone)
PHYS_FORM(acetone,chemical)
PHYS_FORM(acetone,liquid)
DESCRIPTION_COLOUR(acetone,colorless)
DESCRIPTION_SMELL(acetone,distinct)
PROPERTY(acetone,burn,NONE,NONE,easily)
PROPERTY(acetone,evaporate,NONE,NONE,easily)
PROPERTY(acetone,dissolve,water,in,NONE)
ORIGIN(acetone,natural,vehicle exhaust,in)
ORIGIN(acetone,natural,tobacco smoke,in)
ORIGIN(acetone,natural,landfill site,in)
ORIGIN(acetone,natural,plant,in)
ORIGIN(acetone,natural,the environment,in)
ORIGIN(acetone,man-made,NONE,NONE)
ORIGIN(acetone,natural,tree,in)
ORIGIN(acetone,natural,volcanic gas,in)
ORIGIN(acetone,natural,forest fire,in)
ORIGIN(acetone,natural,a product,in)
SYNONYM(acetone,dimethyl ketone)
SYNONYM(acetone,beta-ketopropane)
SYNONYM(acetone,2-propanone)
USE(acetone,NONE,NONE,make,plastic,present)
USE(acetone,NONE,NONE,make,fiber,present)
USE(acetone,NONE,NONE,make,drug,present)
USE(acetone,NONE,NONE,make,other chemical,
                                  present)
USE(acetone,NONE,NONE,dissolve,
                  other substance,present)
```

Most of the information present in the original text has been extracted and normalized:

for example, *flammable* is normalized as **PROPERTY(acetone,burn,NONE,NONE,easily)**. However, form the input ... *as a product of the breakdown of body fat*, the system extract the partial analysis **ORIGIN(acetone,natural,a product,in)**. Such cases are discussed in section 4.

In this section, we have shown how, extending a general parser with limited information (morphological and transformational) and adding specific domain knowledge for the corpora we consider, we were able to obtain a normalization of some knowledge enclosed in the texts. The next section is dedicated to the evaluation of the performances of this system.

4 Evaluation

We decided to perform two kinds of evaluation

- First, we wanted to check if our system performs correctly the extraction of the selected information.
- Second, we wanted to verify the impact of the normalization and the corpus oriented paraphrase modules in the obtained results.

4.1 Performance of the whole system for information extraction

In order to evaluate the results of the information extraction system, we apply the full chain of information extraction on an unseen collection of 30 texts describing toxic substances. Then we associate the output predicates to the corresponding texts and ask each of the five evaluators to compare six pairs of texts/predicates. We ask them to read carefully the texts and to fill a table which covers the different types of information in scope, i.e substance, physical form, colour, odor, synonyms, physical properties, and use. For each topic, they have to express what is missing, superfluous or wrong in the list of predicates, compared to the original texts. We consider one missing answer for each missing information detected by the evaluators. And we consider an incorrect response for each information that had been extracted by the system and that did not correspond to any realization in text. We then compute precision and recall, obtaining the following results:

Precision	Recall	F-score
.96	.65	.77

We obtain a high precision result which could be expected considering our IE methodology. In most of the cases, when the information has been extracted, it is correct. However, most of the problems are a consequence of insufficient coverage of both the extraction grammar (problems with structural ambiguity) and domain-knowledge. The main sources of errors which have been identified during the evaluation comes from :

- Coordination detection problems. For example, from the sentence *Hexachlorobutadiene is also used as a solvent, and to make lubricants, in gyroscopes, as a heat transfer liquid, and as an hydraulic fluid.* the system detects only one "use" of the element: **USE(Hexachlorobutadiene,solvent,as,NONE,NONE)**, because the complex coordination has not been solved.
- Scope of the extraction: from the sentence *Nitrobenzene is used in the manufacture of dyes*, the system extracts **USE(Nitrobenzene,manufacture,in,NONE,NONE)**, because the PP *of dyes* was not expected in the structure of the **USE** predicate.
- Domain-knowledge coverage: form the sentence *Acetone completely miscible in water and soluble in organics.*, the system extract **PROPERTY(Acetone,dissolve,in,organic,NONE)**, because *soluble* is encoded as a property equivalent to *dissolve* in the lexical relations for paraphrasing. However, it should also extract **PROPERTY(Acetone,mix,in,water,NONE)**, but *miscible* was not coded as a possible chemical property adjective.

From the evaluation results, it appears that further developments need to focus on recall improvement. This could be achieved by:

- extending our paraphrase detection module: Some equivalences have not been yet considered. For instance, *take fire* which did not appear in the working corpus, appeared in the test corpus. This expression had not been coded as a possible equivalent of *burn*, therefore expected information about the physical property of burning for a given element is missing when this property is expressed in the text by *take fire*;

- enriching the ontological knowledge of the domain;
- Improving structural ambiguity resolution: Coordination and PP attachment resolution could be improved by the development of more fine-grained semantic and ontological resources.

4.2 Impact of the normalization and corpus oriented paraphrase modules

This second experiment was intended to verify in what extent the normalization and paraphrase detection module affect the results obtained in the previous evaluation. This test was performed taking away from the complete processing chain, the modules described in sections 3.2 and 3.3.2. The results show that we only obtained about 60% of the predicates found in the first version. In other words, without these processing steps, recall decreases in a dramatic way. All predicates found in this second experiment were also found in the first. Missing predicates in the second experiment were the most complex to extract (i.e. USE, PROPERTY, ORIGIN), since they intensively involve reformulations and lexical equivalences.

5 Conclusion

In this paper, we have presented a methodology for extracting information using symbolic methods. Information extraction consists here in normalization of syntactic processing using both deep syntactic and morphological information as well as corpus specific knowledge. As the kind of corpus under consideration is very rich in reformulations, we were able to verify that our system could be used to detect paraphrases in the domain of the corpus. In fact, paraphrase detection can be seen as a side effect of normalization, as utterances conveying similar information are represented the same way. This is an ongoing work but the first results we obtained for information extraction are really encouraging, although many improvements seem to be necessary. We foresee to continue our experiment applying our system on a different collection of texts from the same domain. We also plan to improve the current coverage of our system having in mind the results of the first evaluation.

Acknowledgments

We would like to thank our colleagues Jean-Pierre Chanod, Marc Dymetman, Aaron Kaplan and Ágnes Sándor for their careful reading and helpful comments on this paper.

References

Salah Aït-Mokhtar, Jean-Pierre Chanod and Claude Roux. 2002. Robustness beyond shallowness: incremental dependency parsing. *Special issue of the NLE Journal*.

Regina Barzilay and Kathleen R. McKeown. 2001. Extracting Paraphrases from a Parallel Corpus *Proceedings of the ACL 2001* Toulouse, France.

Ralf Grishman, Catherine Macleod, and Adam Meyers. 1994 COMLEX: building a computational lexicon In *Proceedings of the 15th International Conference on Computational Linguistics (COLING 1994)*, Comlex.

Nobuhiro Kaji, Daisuke Kawahara, Sadao Kurohashi, and Satoshi Sato. 2001. Verb Paraphrase based on Case Frame Alignment *Proceedings of the Workshop on Automatic Paraphrasing*. NLPRS 2001, Tokyo, Japan.

Beth Levin. 1993. *English Verb Classes and Alternations - A Preliminary Investigation*. The University of Chicago Press.

Kathleen R. McKeown, Judith L. Klavans, Vasileios Hatzivassiloglou, Regina Barzilay, and Eleazar Eskin. 1999. Towards Multidocument Summarization by Reformulation: Progress and Prospects. *AAAI/IAAA*.

Carolyn P. Rosé. 2000. A syntactic framework for Semantic Interpretation. *Proceedings of the 1st meeting of the North American Chapter of the Association for Computational Linguistics*. Seattle, Washington.

Celex 2000. http://www.kun.nl/celex/index.html.

Tetsuro Takahashi, Tomoyam Iwakura, Ryu Iida, Atsushi Fujita and Kentaro Inui. 2001. KURA: A Transfer-Based Lexico-Structural Paraphrasing Engine. *Proceedings of the Workshop on Automatic Paraphrasing*. NLPRS 2001, Tokyo, Japan.