Using the DIFF Command for Natural Language Processing

Masaki Murata and Hitoshi Isahara

Communications Research Laboratory,
2-2-2 Hikaridai, Seika-cho, Soraku-gun, Kyoto, 619-0289, Japan,
{murata,isahara}@crl.go.jp,
WWW home page: http://www.crl.go.jp/khn/nlp/members/murata/index.html

Abstract. *Diff* is a software program that detects differences between two data sets and is useful in natural language processing. This paper shows several examples of the application of *diff*. They include the detection of differences between two different datasets, extraction of rewriting rules, merging of two different datasets, and the optimal matching of two different data sets. Since *diff* comes with any standard UNIX system, it is readily available and very easy to use. Our studies showed that *diff* is a practical tool for research into natural language processing.

1 Introduction

Diff, a software program that detects differences between two sets of data, has numerous applications in natural language processing. In this paper, we first describe *diff* and then give several examples of how *diff* can be used in natural language processing. These examples include the detection of differences, the merging of two sets of different data, the extraction of rewriting rules, and the best matching of two different sets of data.[1]

We summarize the value of this paper below.

– Since *diff* is one of the basic programs that comes with UNIX systems, it is readily available and very easy to use. As this paper shows, *diff* is also applicable to various kinds of studies in the field of natural language processing. This combination makes the program particularly valuable.
– Paraphrasing has been become an active area of study [2]. Section 3.2 shows that we can use *diff* to examine the differences between a spoken language and the corresponding written language and in acquiring the rules for the transformations between the spoken and written languages. *Diff* is not only useful for such studies, but is also applicable to other kinds of studies in the field of paraphrasing. This paper will provide a basis for such further application of *diff*.
– In general, *diff* is used to detect differences between two data. However, it can also be used to merge data and to match data optimally. Section 4 covers

[1] This papar is an English rough translation of our papers [1,3].

several interesting studies of the use of *diff* in merging and best-matching tasks. These examples are the alignment of a paper and a corresponding presentation and a question-answering system. This paper thus has originality in terms of topics researched and approach to the research.

2 Diff and Mdiff

In this section, we describe *diff*. *Diff* is a software program of Unix systems that is used to compare files. The program displays line-by-line differences between a pair of text files while retaining the order of the data. For example, suppose we have the following two files:

```
File 1:              File 2:
I                    I
go                   go
to                   to
school.              university.
```

When we give these data to *diff*, the difference is displayed in the following way.

```
< school.
> university.
```

Diff has a *-D* option, which is very useful. When we use *diff* with this option, common parts as well as differences are displayed. That is, files can be merged by using this option. However, the output of *diff -D* is in a form which is used for C preprocessor such as "Ifdef" and this is difficult for people to read. Therefore, in this paper, we give the differences in the following way:

```
;===== begin =====
(The parts which only exist in the first file)
;-----------------
(The parts which only exist in the second file)
;=====  end  =====
```

where, "`;===== begin =====`" indicates the beginning of the differences, "`;===== end =====`" indicates the end of the differences, and "`;-----------------`" indicates the boundary between the two sets of data. In this paper, we refer to *diff* in the case where the files are merged by using the `-D` option and the differences are displayed in the above form as *mdiff*.

When we give our earlier pair of files to *mdiff*, we obtain the following result.

```
I
go
to
;===== begin =====
school.
;-----------------
university.
;=====  end  =====
```

"I go to" matchs while "school" and "university" are differences. The output of *mdiff* is easy to examine and understand because it also displays, unlike *diff*, the common parts.

We can reproduce the two original files from the output of *mdiff*. When we take the common part and the upper part of the differences, we obtain the content of the first file. When we take the common part and the lower part of the differences, we obtain the content of the second file. We can reproduce all of the original data in this way.

Since *mdiff* only displays the common part of the data once, it is able to reduce the amount of data. Since it is possible to fully reproduce the original data from *mdiff*'s output, we are able to say that *mdiff* compresses the data while retaining the original information.

Since the output of *diff* is difficult to read and the output of *mdiff* contains all information output by *diff*, we use *mdiff* for our explanations in the following sections. In the following sections, let's look at some actual examples of the application of *mdiff* to natural language processing.

3 Detecting differences and acquiring transformation rules

In this section, we describe the studies where we used *mdiff* to detect differences and then acquired transformation rules from the differences. In more concrete terms, we give the following two examples.

- Detection of differences between the outputs of multiple systems
- Examination of differences to acquire transformation rules

3.1 Detection of differences between the outputs of multiple systems

We have been using multiple morphological analyzers to improve the quality of the results of morphological analysis.

Suppose that we use two morphological analyzers to analyze "We like apples."; the results are as follows.

```
System 1                 System 2
We       Noun            We       Noun
like     Verb            like     Preposition
apples   Noun            apples   Noun
```

The word, "like" is ambiguous in terms of part of speech (POS) and can take "Verb" or "Preposition". The POS of "like" in the above sentence is "Verb" and the analysis by System 2 was wrong. Here, when we give the two results to *mdiff*, we obtain the following results.

```
We      Noun
;===== begin =====
like    Verb
;-----------------
like    Preposition
;=====  end  =====
apples  Noun
```

Mdiff makes it easy for us to detect differences between the results produced by multiple systems. In this case, we notice that there are differences in the line for "like". Here, we determine beforehand that a worker corrects the outputs and judges which is correct when such differences are detected. When the output of the first system is correct he does nothing and when the output of the second system is correct he marks the beginning of ";——————" with an "x". When we do so, we can automatically select the better result of each difference from the results marked by the worker. We simply delete the lower part of the difference where there is no "x" and delete the upper part of the difference when there is an "x". As a result, we produce a more accurate overall result than either of the two original results. There are many cases when both parts of a difference are incorrect. In such cases, we manually rewrite the data correctly above ";——————".

When we use this method, the only incorrect data that is not corrected are the data on which both the systems have performed exactly the same wrong analysis. Many errors can thus be corrected. What we want to say here is that we must prepare multiple systems that have different characteristics. If we use two systems that often produce the same wrong analysis, we will fail to correct a large number of errors.

Diff3, which is another unix program, is available for analysing the outpus of three systems. *Diff3* detects differences among three files.

Although we have given an example of morphological analysis, we can also use *mdiff* to detect differences between the results of other forms of analysis as long as the results are expressed in a line-by-line form.

Although we have described a study where we detected differences between the outputs of multiple systems, we are also able to detect and correct errors in tagged corpora by comparing the tagged corpora and the results of analysis of the corpora by a certain system.[2]

3.2 Examining differences and acquiring transformation rules

In this section, we describe our study of the use of *mdiff* to detect differences between a spoken language and the corresponding written language. In this study, we used the alignment between data on the spoken language and on the written language to examine the differences between the spoken and written languages on the basis of the results of *mdiff*. We thus acquired rules for transforming the

[2] There are studies on the correction of errors in tagged corpora [5].

Table 1. Examples of written data and the corresponding data in spoken form

The written-language sample
In
this
paper,
we
describe
the
meaning
sort.
In
general,
sorting
is
performed
by
using

The spoken-language sample
Today
I'd
like
to
describe
uh
the
meaning
sort.
In
general,
sorting
is
done
by

spoken language to the written language along with the inverse transformation rules. We used presentations at academic conferences as examples of the spoken language and used papers which had the same content as the presentations as examples of the writtten language.

For example, we supposed that data in the spoken and written languages are given as in Table 1.[3] The data in the table are transformed so that each line has one word before we apply *mdiff*. Applying *mdiff* to the data produces the results shown in Table 2. By extracting the differences from these results, we obtain the results shown in Table 3.

The results show us that "uh" is inserted in the spoken language and that "performed" can, in this case, be paraphrased as "done". We can use *mdiff* to detect differences between the spoken language and the written language and then examine them linguistically. We can also consider the detected differences as rules for the transformation between written and spoken languages. For example, the occurrence of "uh" can be regarded as indicating a rule that we may insert "uh" into the written-language data as part of transforming it into the spoken language. The difference in terms of whether "performed" or "done" is used can be regarded as a rule that we transform "performed" to "done" when we transform data from the written to the spoken language. We thus find that we are able to use *mdiff* to detect the transformation rules.

In this section, we used data in the written and spoken language. However, we can study a variety of data in a similar way. For example, when we apply *mdiff* to written texts before and after spelling and grammar are checked, we can find out

[3] Although we have given English sentences in the table, we actually performed these experiments on Japanese-language materials.

Table 2. The result of applying *mdiff* to the written-language sample and the corresponding spoken-language sample

;===== begin =====
In
this
paper,
we
;————
Today
I'd
like
to
;===== end =====
describe
;===== begin =====
;————
(Contined in the right-hand column)
uh
;===== end =====
the
meaning
sort.
In
general,
sorting
is
;===== begin =====
performed
;————
done
;===== end =====
by

Table 3. Extraction of differences

The written-language data	The spoken-language data
In this paper, we	Today I'd like to
	uh
performed	done

how we should modify the sentences to eliminate spelling and grammatical errors and thus acquire rules for checking spelling and grammar. When we use *mdiff* on texts and the results of their automatic summarization, we can clearly see how the texts were summarized and can thus acquire the rules used in automatic summarization. In other cases of paired textual data, too, we will be able to apply *mdiff* to examine the data in the various ways that such pairs can be obtained and to obtain the rules for transformation.

4 The merging and best matching of two different sets of data

In this section, we describe studies of the use of *diff*'s function of optimally merging data.[4] We describe the following two examples.

– The alignment of presentations and corresponding papers
– A question-answering system using the function of finding the the best match

[4] In *diff*, best matching is performed by maximizing the extent of the common parts.

```
<Chapter 1>
(the contents of Chapter 1)
</Chapter 1>
<Chapter 2>
(the contents of Chapter 2)
</Chapter 2>
<Chapter 3>
(the contents of Chapter 3)
</Chapter 3>
```

Fig. 1. Structure in the paper

```
;===== begin =====
<Chapter 1>
(content of the paper only)
;————————
(content of the presentation only)
;===== end =====
(content of the paper and presentation)
;===== begin =====
(content of the paper only)
;————————
(Contined in the right-hand column)
```

```
(content of the presentation only)
;===== end =====
(content of the paper and presentation)
;===== begin =====
</Chapter 1>
<Chapter 2>
(content of the paper only)
;————————
(content of the presentation only)
;===== end =====
```

Fig. 2. Results of applying *mdiff* to a paper and a corresponding presentation

4.1 Alignment of a paper with the corresponding presentation

In this section, we align a paper with the corresponding presentation [6]. The paper and corresponding presentation is the same example as was used in describing the acquisition of the rules for transformation from written to spoken-language data (Section 3.2). The presentation was made at an academic conference and the paper corresponds to this presentation. When we are able to align each part of a presentation with a corresponding part of the paper, this form of alignment is very useful. For example, when we listen to a presentation, we are able to refer to the part of the paper that corresponds to the part of we are now listening. When we read a paper, we are able to refer to the part of the presentation that corresponds to the part of the paper we are now reading [6]. In this section, we consider the use of *mdiff* to align a paper with the corresponding presentation.

Here, let's try to determine the parts of the presentation to which each chapter of the paper corresponds by using *mdiff*. We suppose that the content is laid out in the same order in the paper and in the presentation. In advance, we place symbols such as `<Chapter 1>`, as shown in Figure 1, into the paper. This is so that we are easily able to recognize the chapters of the paper. By applying *mdiff* to the data set that consists of a presentation after both have been transformed so that each line has one word, we obtained the results shown in Figure 2. Next,

```
<Chapter 1>
(content of the presentation only)
(content of the paper and presentation)
(content of the presentation only)
</Chapter 1>
<Chapter 2>
(content of the presentation only)
```

Fig. 3. Results of insertion of information on sections in a presentation

we obtained the results shown in Figure 3 by eliminating the upper parts of the differences, i.e., those that correspond to the paper, and leaving symbols such as `<Chapter 1>` in place. The symbols such as `<Chapter 1>` are only inserted in the data of the presentation. We are then able to recognize the chapter of the paper that corresponds with a given part of the presentation.

To put this simply, we place information to indicate chapters in the data of presentations. We then use the merging function of *mdiff* to match the parts of the paper and presentation. Then, by eliminating the content of the paper, we are left with information on the chapters. We can easily align a paper with a corresponding presentation at the level of chapters by using *mdiff*.

4.2 A question-answering system using best matching

In this section, we describe our question-answering system that utilizes *mdiff*'s best-matching function [2,4].

A question-answering system receives a question and outputs an answer. For example, when "Where is the capital of Japan?" is given, it returns "Tokyo". When we consider now much knowledge is in written natural language form, we see that the system only has to match a given question with a sentence (a knowledge sentence) that includes the knowledge then select the word in the sentence that corresponds to the interrogative pronoun of the question sentence. For example, in the case of the previous question, the system detects this knowledge sentence in a knowledge database: "Tokyo is the capital of Japan." "Tokyo" is output as the answer since corresponds to the interrogative pronoun in the sentence. Here, we consider the use of *mdiff* in answering questions.

Firstly, we transform the interrogative pronoun in the question sentence into X and change "?" into "." at the end of the sentence. We then obtain "X is the capital of Japan.". We thus obtain "Tokyo is the capital of Japan." from the knowledge base. We obtain the results in Figure 4 (a) by applying *mdiff* to the two sentences. We consider the part that is paired with X in the differences as the answer and correctly obtain "Tokyo".

Even if there are differences between the compared sentences, we can still use *mdiff* to correctly extract the answer. For example, suppose that the knowledge sentence is "Tokyo is the capital in Japan." In this case, the result of *mdiff* is

```
;===== begin =====
X
;-----------------
Tokyo
;=====  end  =====
is
the
capital
of
Japan.
```

(a) Case 1

```
;===== begin =====
X
;-----------------
Tokyo
;=====  end  =====
is
the
capital
;===== begin =====
of
;-----------------
in
;=====  end  =====
Japan.
```

(b) Case 2

Fig. 4. Results of *mdiff* in the question-answering system

as in Figure 4 (b). Although the number of differences increased, the part that corresponds to X remains "Tokyo" and the correct answer has been detected.

The question-answering system we have proposed repeats to transform the question sentence and knowledge sentence so that the two sentences are as similar as possible. It then extracts the answer by matching the sentences when they are at this point. Since we use the similarity between two sentences in that time, we have to define the degree of the similarity between sentences. Since we can recognize the common parts and the different parts by using *mdiff*, we can calculate the degree of similarity as (the number of characters in the common part)/(the number of all characters).[5] Here, if we suppose that we have a rule for transforming "in" to "of", we match the data after transforming "Tokyo is the capital in Japan." to "Tokyo is the capital of Japan." and we are able to obtain the answer more reliably by decreasing the number of differences.

5 Conclusion

In this paper, we describe a lot of examples of the use of *diff* in various problems of natural language processing. In Section 3.1, we describe how we apply *diff* to a system that combines multiple systems and thus obtain higher precisions than from any of individual systems. Studies on combination of systems are very well known. We describe how such combination can easily be performed by *diff*. In Section 3.2, we showed that *diff* is easily able to examine the differences between a spoken and written language and acquire rules for the transformation between

[5] Here, we use *mdiff* to calculate the degree of similarity of the sentences. *Mdiff* is also useful in this way.

the spoken and written language. We handled the paraphrasing of written language into spoken language. However, the study on paraphrasing covers a wide range: automatic extraction of compressed sentences, sentence polishing-up to modify sentences correctly, and transformations from difficult sentences to plain sentences. In these studies, too, it will be easy to apply *diff* to various forms of examination and to the extraction of various transformation rules. This paper will become a basis for studies of paraphrasing, an area in which there has been increasing activity. In Section 4, we showed examples of the merging of data and the optimal matching of data. Since *diff* is generally used to detect differences, Section 4, which shows applications of *diff* to merging and matching, presents a new and original approach. In the section, we showed that it is easy to apply *diff* to interesting studies of two kinds, the alignment of a paper with a corresponding presentation and a question-answering system.

In this paper, we have described many examples of the application of *diff* to natural language processing. We hope that *diff* will be applied to an ever-widening range of studies.

Acknowlegement

We were given data for the experiments on alignment of a paper with a corresponding presentation by Kiyotaka Uchimoto of the Communications Research Laboratory.

References

1. Masaki Murata and Hitoshi Isahara. Nlp using diff. *IPSJ-WGNL 2001-NL-144*, 2001. (in Japanese).
2. Masaki Murata and Hitoshi Isahara. Universal model for paraphrasing: Using transformation based on a defined criteria. In *NLPRS'2001 Workshop on Automatic Paraphrasing: Theories and Applications*, 2001.
3. Masaki Murata and Hitoshi Isahara. NLP using DIFF — use of convenient tool for detecting differences, MDIFF —. *Journal of Natural Language Processing*, 9(2), 2002. (in Japanese).
4. Masaki Murata, Masao Utiyama, and Hitoshi Isahara. Question answering system using syntactic information. 1999. http://xxx.lanl.gov/abs/ cs.CL/9911006.
5. Masaki Murata, Masao Utiyama, Kiyotaka Uchimoto, Qing Ma, and Hitoshi Isahara. Correction of the modality corpus for machine translation based on machine-learning method. *7th Annual Meeting of the Association for Natural Language Processing*, 2001. (in Japanese; an English translation of this paper is available at http://arXiv.org/abs/cs/0105001).
6. Kiyotaka Uchimoto, Chikashi Nobata, Kimiko Ohta, Masaki Murata, Qing Ma, and Hitoshi Isahara. Segmenting the transcription of a talk by aligning the transcription with its corresponding paper. *7th Annual Meeting of the Association for Natural Language Processing*, pages 317–321, 2001.