A Multilingual Usage Consultation Tool Based on Internet Searching -More than a Search Engine, Less than QA- *

Kumiko Tanaka-Ishii
Graduate School of Information Science and Technology
University of Tokyo
7-3-1 Hongo, Bunkyo-ku
Tokyo, Japan
kumiko@mist.i.u-tokyo.ac.jp

Hiroshi Nakagawa
Information Technology Center
University of Tokyo
7-3-1 Hongo, Bunkyo-ku
Tokyo, Japan
nakagawa@dl.ic.u-tokyo.ac.jp

ABSTRACT

We present a usage consultation tool, based on Internet searching, for language learners. When a user enters a string of words for which he wants to find usages, the system sends this string as a query to a search engine and obtains search results about the string. The usages are extracted by performing statistical analysis on snippets and then fed back to the user.

Unlike existing tools, this usage consultation tool is multilingual, so that usages can be obtained even in a language for which there are no well-established analytical methods. Our evaluation has revealed that usages can be obtained more effectively than by only using a search engine directly. Also, we have found that the resulting usage does not depend on the search engine for a prominent usage when the amount of data downloaded from the search engine is increased.

Categories and Subject Descriptors

I.7.1 [**Document and Text Editing**]: Languages; H.4.m [**Information Systems**]: Miscellaneous

General Terms

Design

Keywords

Usage consultation, question answering, text mining.

1. INTRODUCTION

With the recent developments in communication media and a growing awareness of foreign cultures, people have become more exposed to foreign languages than at any other time in recent history. More people are learning foreign languages, and language tools are increasingly important. In this paper, we deal with the issue of how a person studying a foreign language can learn word usage in that language.

For language learners stuck on problems of language usage, dictionaries based on fine distinctions and abstracted usages have been the main means of obtaining answers for hundreds of years. Dictionary revisions, however, are few and far between, and users often cannot find explanations concerning current usages or specialized terms in dictionaries. Another way to search for information on usage is to use the KWIC (Key Word in Context) tool. Recent advances in the full-text search algorithms originally proposed by Manber [11] now allow users to instantly consult text data encompassing several gigabytes. The problem, however, is that large bodies of data are not yet readily available in languages used less commonly in international situations.

Given that the World Wide Web has become the largest electronic text body available in any language, one modern solution is to use a search engine. For example, if a learner wants to find out which verb is used with "jet lag" in English, he can simply type "jet lag" into a search engine and obtain usages such as "avoid jet lag" and "recover from jet lag" among various snippets of information. Similarly, search engines can be used to find out how to use articles correctly: for instance, in French, non-natives will have problems with usages such as: "*revenir du/de Paris*" (here, it's de), compared with '*revenir du/de Japon*" (here, it's du).

Unfortunately, the problem here for users is that the search results are simply lists of individual usages. Of course, the search engine ranks the usages, but they are *not ranked in terms of linguistic relevance.* Also, using materials from the Web for language learning can be risky, since there are sites where language is not used in a standard style. One way to verify such usages is to scan through several pages and statistically determine the most prominent usage. In the case of the verb used with "jet lag", a user can see that "avoid" appears many times by scanning through multiple pages of search results.

To automate this task of scanning and summing, we created a tool, called Kiwi, that can read a large number of pages on behalf of the user and statistically accumulate usages from snippets of those pages.

The idea of creating such a tool to facilitate usage lookup is not new. The earliest such existing tool is WebCorp [17]. Upon receiving a query, Webcorp sends it to a search engine

*

WWW 2005, May 10-14, 2005, Chiba, Japan.
ACM 1-59593-046-9/05/0005.

and totals the results. This tool is only available in English, however, and it is not readily applicable to languages without word segmentation. Other interesting examples are Google Fight [6] and Google Duel [13], in which the user enters two different phrases and the tool suggests which is more commonly used according to the frequency reported by Google. Although these two sites allow multilingual consultation, the range of flexibility is limited to the "duel" of two given phrases. Additionally, the effectiveness of this idea of summing up search engine results in the usage lookup context has not been verified.

In contrast to these existing tools, Kiwi allows flexible entry, making it language-independent. This feature is implemented by using our language-free candidate extraction methods based on techniques originally proposed in the term extraction domain of natural language processing (NLP).

Interestingly, Kiwi can be applied not only for language usage consultation but also be used as a "pseudo" question-answering (QA) tool. Of course, the input to our tool is a mere query, and it therefore cannot directly receive questions, as is done in QA. We can obtain answers, however, by entering a query in the form of an answer, such as "* is the president of France", or "* is the capital of Tonga". Then, the system will look for strings that can fill the * part, as if looking for usages. Thus, Kiwi can be situated in between a search engine and a question-answering tool, having the feature of summing up search engine results.

This feature is also utilized in recent QA systems. For example, several QA systems utilize search engine results to obtain answers by summing the results [10][1][3]. In such studies, the interest was in the quality of the answers, so the performance was evaluated in terms of the whole results of the QA procedure. In contrast, the statistical behavior of the summing process has not been thoroughly discussed.

Thus, another contribution of this paper is to provide a fundamental analysis of the statistical behavior of summing up search engine results. We first discuss questions such as the effectiveness of summing up results, the number of snippets needed to obtain the results, and the effectiveness for a user trying to acquire a target usage by looking at summed results. Another important question is the statistical behavior when different search engines are used. The hypothesis given here is that even for search engines using different ranking methods, when a large quantity of search results is obtained and summed up, the engines' results will converge to the same results. We thus discuss the commonness of usages acquired from different search engines.

In the next section, we describe the overall design of Kiwi. Then, we explain our language-independent data analysis method, especially for candidate extraction. In Section 4, we present example usages obtained using Kiwi. Then, from section 5 on, we evaluate the usability of Kiwi from various points of view: through automatic experiments on collocations verifying the accuracy of usage extraction, through user evaluation of the tool's speed and the number of actions required, and finally, by comparing the results obtained by different search engines.

2. OVERALL SYSTEM DESIGN

Kiwi is software situated between a user and a search engine (Figure 1). Kiwi passes through the following stages in one cycle of usage consultation:

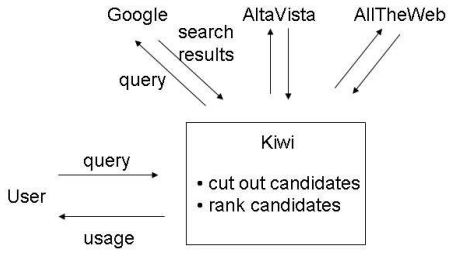

Figure 1: Overall design of Kiwi

Figure 2: The Kiwi site

1. Receive an input query from a user.
2. Send the query and obtain a fixed number of search engine results that match the query.
3. Extract all fixed-length strings that might include candidates.
4. Cut the candidates from these strings.
5. Rank the candidates.
6. Display the results.

The essential procedure of Kiwi is the totalization of the usages triggered by a query, which corresponds to stages 3-5. Kiwi can be implemented as simple client software communicating directly with a search engine server. Currently, however, it has been developed on a server available at [16].

Figure 2 shows a Kiwi site with a query on the Chinese expression of "how * is!" (to look for adjectives in the position of *). In the topmost part is a query box, where the user enters a query; in the next line are two pulldown menus for choosing a language and the number of snippets to be downloaded. The lower half contains the lists of resulting usages. Each line shows a possible usage acquired from the query and can be clicked to return to the original, individual snippet. This is pasted on the right-hand side of Figure 2. With this clicking function, the user can view how the usage actually appears in a longer context.

Compared with previous works, such as [17] and [6], Kiwi has two distinctive features: the query flexibility, and its multi-lingual capability. First, regarding flexibility, every query is constructed by a user based on a pattern match. Basically, there are three patterns:

Wild card '*' can be used to replace a missing word or string. For example, "human *", "* sans fil", or "fed * with" can be entered into Kiwi.

Comparison '/' allows users to compare two or more phrases and prioritize them depending on which is the most relevant. This can be expressed as (A/B/C...).

And search '+' can be used for narrowing the search domain. For example, "* Bush +American president" or "chateau * +vin France" can be entered into Kiwi.

The cognitive background of this design is as follows. When a user has a question regarding language usage, he often cannot clearly remember the complete expression but still has a few clues that can be utilized to access language data. Such vague recollections typically follow two major patterns: in one, the user lacks the information needed to complete part of a phrase, while in the other, he is uncertain in choosing the correct word from among several candidates.

Second, as noted above, Kiwi is language independent. In any language, there is the problem of updating conventional resources to include current language usage. Conventional dictionary revisions are few and far between. Such cases are emphasized for languages that are not widely used internationally. The available dictionaries are usually older, the available corpora are too small for KWIC, and the cycle of revising the data is likely to be longer. Therefore, students learning a foreign language other than English often find that the available language resources are limited, and that they must rely more on search engine results. This is the main reason why we designed Kiwi to be independent of language; that is, Kiwi uses neither language-dependent algorithms nor a language dictionary for analyzing search engine results.

In the next section, we describe the essential procedure inside Kiwi: candidate processing.

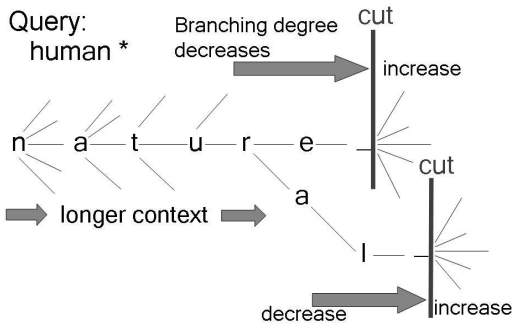

Figure 3: Concept behind the language-independent candidate extraction methodology

3. CANDIDATE PROCESSING

3.1 Candidate Extraction

When a wild card is used at the beginning or end of a query, Kiwi needs to cut out the relevant chunk[1]. As the method of processing the beginning and end of a query is symmetric, candidate extraction to obtain the end part of the query is discussed here.

The easiest way is to cut out a string, or a group of words, of a fixed length. Fixed-length strings are commonly used in the KWIC system for non-segmented languages. One drawback of using fixed-length strings is that it hinders the aggregation of statistics obtained from data. For example, although "with" is often found after "be acquainted", all occurrences of "with" cannot be put together if the length is fixed at 10 characters, because there are a variety of strings that can come after "with." Therefore, we want a candidate that has the correct border.

Intuitively, we would expect a candidate to have the following properties:

A It occurs frequently.

B It has moderate length.

C It is succeeded by various kinds of characters.

A concerns the relevance of the candidate (as will be discussed in the next section). On the other hand, **C** essentially expresses information about word boundaries. **B** can be used for either objective. Therefore, we have tried to implement candidate extraction by combining **C** and **B**.

C leads to the idea of looking at the branching degree at a language chunk border, which has been discussed in the context of automatic term-recognition methods [5][12]. In these works, the branching notion was used as part of the extraction evaluation function for detecting a collocation border. These methods, however, are *word* based and presume the use of taggers for non-segmented languages.

The above observations on *word* sequence can be generalized to a character sequence as follows. If the branching degree at each character inside a word is observed, it is seen to monotonically decrease according to the offset length. This is a natural result, because the longer the preceding character n-gram, the longer the preceding context. On the other

[1]When it is used in the middle, the boundary can be straightforwardly determined.

Table 1: Live examples of Kiwi

language	Input String	Result Examples (translation)		
English	wireless*	network	communications	internet
English	Matrix *	reloaded	revolutions	software
English	* Syndrome	down	chronic fatigue	severe acute respiratory
English	Arnold *	Schwarzenegger	Palmer	Kling
English	Harry Potter*	and the Chamber of Secrets	and the Sorcerer's Stone	and the Order of the Phoenix
French	tour*	de france	du monde	eiffel
French	*sans fil (wireless)	réseaux (network)	des réseaux (network)	téléphone (telephone)
Japanese	無線* (wireless)	LAN (LAN)	LAN カード (LAN card)	通信 (networking)
Japanese	東京* (Tokyo)	商工会議所 (Chamber of Commerce and Industry)	大学 (University)	ディズニーランド (Disney Land)
Japanese	*純一郎 (Jun-ichiro)	小泉 (Koizumi)	内閣総理大臣 小泉 (Prime Minister Koizumi)	
Chinese	无线 * (wireless)	网络* (network)	电管理 (radio management)	局域网 (LAN)
Chinese	*烤鴨 (duck)	北京 (Beijing)	全聚德 (the name of restaurant)	杨家吊炉饼 (kind of Chinese bread)
German	*Bach	Sebastian	Johann Sebastian	Carl Philipp Emanuel
Korean	*김치 (kimchi)	포기 (Pogi kimchi)	배추 (Chinese cabbage)	층각 (Japanese radish)
Spanish	Real* (Spanish soccer team)	Madrid	federación española de	Sociedad

hand, the average branching degree at the point of a word border becomes greater, as the complexity increases, once it is out of context. This suggests that the word border can be detected by focusing on the differentials of the branching degree. This concept is visualized in Figure 3.

Based on this concept, we devised a simple *string*-based method for cutting out candidates even from non-segmented language data. Most importantly, this method is language independent. With X as a string, let X_i be the head i characters of X, and let C_i be the number of *kinds* of characters at the $(i+1)$th position of all strings with the prefix X_i. We extract X_i as a candidate when

$$C_i > C_{i-1}. \quad (1)$$

Note that this method integrates the notion of the length (condition **B**) at the same time, because the decreased branching degree puts constraints on the length.

To facilitate this procedure, after obtaining snippets, Kiwi transforms the data into a tree by counting the frequencies and branching degrees. Then, using the method described above, Kiwi cuts out candidates by traversing the tree. Related to this approach in totalizing snippets, Grouper [18] transforms snippets into a suffix tree in order to classify documents by grouping snippets semantically. Their method will be applicable to our simple tree approach in the future, for the purpose of organizing the resulting usages further on from a contextual point of view.

Note that there are cases where this method will always fail to extract candidates: i.e., when a candidate occurs only once. More generally, the extraction with our method tends to fail when a candidate seldom occurs. For example, if there are only two occurrences of "international" and "intention", then "inte" is output as a word. To cope with such cases, we can use a stop word, such as the space symbol, as a delimiter for words in segmented languages. The treatment of stop words was actually considered for the actual system described in [16][2]. Nevertheless, we performed all of the experiments described in §5 without special processing of symbols.

3.2 Candidate Ranking

After candidates are obtained, Kiwi ranks them. Here, the most important information is the frequency (condition **A**). The length is also of concern, though, because short strings obviously occur more frequently than longer ones (condition **B**). Therefore, a bias towards longer candidates should be included in the evaluation function. The question is how best to incorporate these two features into an evaluation function formula.

We decided to empirically choose the evaluation function. With X as the candidate, $|X|$ as its length, and $freq(X)$ as its frequency, we used the following as our evaluation function:

$$F(X) = freq(X)log(|X| + 1). \quad (2)$$

[2]This was the case even though our segmentation can be effective, because it can output word chains as candidates.

Table 2: Top 10 AltaVista results for 'wireless'

1st	Wireless Phone -ATTWireless.com
2nd	Wireless GSM
3rd	MSN for Wireless Electronics
4th	Wireless Phone Specials
5th	Point Wireless has AT&T
6th	Cingular Wireless
7th	U.S. wireless carrier
8th	the wireless plan
9th	Welcome to Verizon Wireless
10th	Guide for Wireless Network

We chose this function because it outperformed all other functions that we considered in the evaluation described from §5 on.

4. KIWI EXAMPLES

Before moving on to our verification on Kiwi's performance, we consider some actual examples of how Kiwi can be used (Table 1). These examples are related to seven languages. Some involve the translation of "wireless network", while others show VIP names, film titles, food names, and so on.

Though we found some data bias towards Internet-related topics (e.g., wireless LAN *cards* being highly ranked because of advertisements), the examples directly reflect our times: Arnold Schwarzenegger watching Harry Potter and the Sorcerer's Stone and The Matrix, eating kimchi and Peking duck, and listening to Bach. Kiwi here is functioning as both a simple question-answering tool and a usage consultation tool.

What we want to emphasize here is that such an overall view cannot be obtained through the direct use of a search engine by the user. Table 2 shows the AltaVista search results for "wireless". (Kiwi currently uses AltaVista as its mother search engine because of its indexing strategy). These results are badly affected by noise. By comparison, Kiwi gives far clearer results: *network*, *communications*, and *Internet*.

5. EVALUATION FOR USAGE

We first evaluated Kiwi's basic performance by using collocations. Our evaluation was done in four stages:

1. Obtain a set of collocations, with each more than three words in length. These collocations were obtained from language-learning books and sites.

2. For each collocation, replace the head, middle, or tail part with a wild card. In the following, these are called *queries*, and the replaced words are called *answers*. Sample queries include "is sick in *" (*bed* is the answer), "catch * with" (*up*), and "il vaut *" (*mieux*, in French, means "it had better"). The resulting queries amounted to 300 in English and 100 each in Japanese and French. The results do not include queries with multiple answers, such as the fact that "in comparison *" can be filled with "to" or "with".

3. For each query, download data matching the query from AltaVista (only snippets). The maximum data quantity was set at 1000 matches for each query.

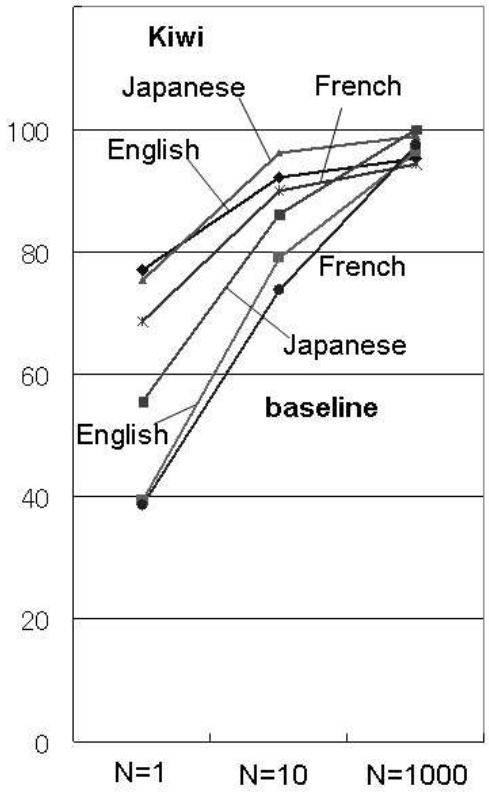

Figure 4: Accuracies for Kiwi and the baseline when the top N candidates were verified

4. Calculate the rate of answers appearing among the top N candidates.

For comparison, a baseline was calculated similarly by looking at the query matches in descending order of the search engine results. The rates were counted with respect to an exact match for the Kiwi results, whereas the baseline results were counted as correct when the corresponding string *included* the required answer.

5.1 Accuracy

Figure 4 shows the Kiwi and baseline results for collocations. The horizontal axis indicates different values of N, whereas the vertical axis indicates the accuracy. Each line corresponds to a different language. The three upper lines correspond to results obtained from Kiwi, whereas the three lower lines show results for the baseline. Each plot corresponds to the results averaged for the head, tail, and middle (i.e., the averaged results of 900 queries for English and 300 queries for French and Japanese).

When N=1 (where only the top-ranking candidate is verified), the Kiwi performance was twice as good as that of the baseline. Also, more than 90% of the Kiwi results included the required answers among the top ten, which was superior to the baseline results. As this tool is intended for human use, this result of answers being included in the top ten is important. We can conclude that the Kiwi process of totalization is effective.

We can also see that Kiwi's performance depends on the availability of data. Data is the most abundant in English,

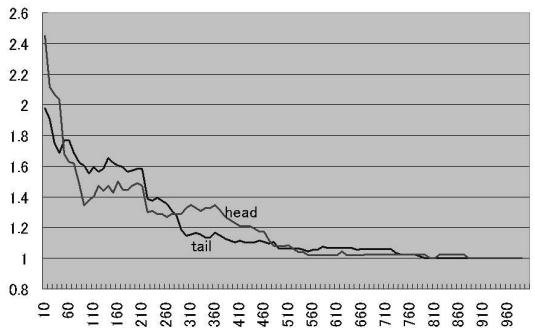

Figure 5: Ranking shift of answers for each downloaded data quantity

then in Japanese, and lastly in French (see [14]). The number of matches for a query was around 600 to 700 in English, 400 to 500 in Japanese, and only about 200 in French. The performance when N=1 in each language exhibits such differences.

There were two cases where Kiwi could not provide the required answers from among the candidates. First, if the answer did not occur frequently enough on the web, Kiwi could not extract it. This defect is caused not by Kiwi itself, but rather by the query not being appropriate for the evaluation, or by the Web not having enough data concerning the query. The error rate attributable to such cases equals (100 - plot of baseline when N=1000)%; this is because, as we collect at most 1000 matches, the plot for the baseline at N=1000 forms the upper bound of the accuracy.

The difference between the Kiwi and baseline plots for the same language at N=1000 indicates the error caused by candidate extraction. This difference was only 1% in English and Japanese but 5% in French, where less data was available as compared with English. This demonstrates the performance of our candidate extraction method using the branching factor. We can say that it was successful in most cases. On the other hand, the difference between the plots at N=1 and at N=1000 for the same line of the Kiwi results indicates the error due to the Kiwi's ranking method of formula (2). This amounts to 10 to 30%. In our future work, we plan to revise the evaluation function used in the ranking process so that the rate of availability of required answers can be pushed higher.

5.2 Data Quantity

Using the same queries, we observed the ranking changes of the answers according to the number of snippets downloaded from the search engine. Figure 5 shows the results. The horizontal axis indicates the number of snippets, while the vertical axis indicates the rankings of the answer. The two lines correspond to the average rankings of the answers for the head and tail parts for English.

We can see that the more snippets are downloaded, the better (lower) the ranking becomes. Finally, it converges to 1.0, meaning that the answer acquired the highest ranking among all other candidates. This happens at around 900 snippets. When small numbers of snippets are downloaded, the ranking still oscillates, and the behavior of Kiwi is unstable.

To download and process 900 snippets, the user would have to wait around 30 seconds on average, and majority of the time is used for downloading. As this is definitely slower than just using a search engine, we are currently modifying the system so that the analysis results are immediately shown for a small number of snippets, then dynamically changed along with the increase in the downloaded quantity of data.

1. How many meters is 1 yard?
2. In which season can we see Orion in Japan, in summer or in winter?
3. What does PDA stand for?
4. Complete the following sentence: "The last shogun of the Tokugawa era was ... th shogun Tokugawa ...".

Figure 6: TREC-like question examples used in the user evaluation

Table 3: User evaluation of Kiwi vs. AltaVista

Required time (mins)		
	Average	Deviation
Kiwi	1.01	0.77
Search engine	1.40	1.18
Number of clicks		
	Average	Deviation
Kiwi	3.40	2.86
Search engine	7.04	6.24
Confidence (1 low — 5 high)		
	Average	Deviation
Kiwi	4.64	0.67
Search engine	4.06	1.11
Accuracy (percentage)		
	Average	Deviation
Kiwi	94.3%	15.2%
Search engine	82.1%	35.5%

6. USER EVALUATION

We next conducted an experimental user evaluation of Kiwi, which had two goals. The first was to see how effective Kiwi could be in actual use. Second, we sought to observe how reasonably Kiwi could be used as a "pseudo" QA system. As explained in §1, our tool can be used for QA tasks by entering answer phrases with the answers replaced by wild cards. This feature is also difficult to evaluate automatically[3]. Consequently, we tested the performance of Kiwi through this user evaluation.

We collected 20 Japanese students from the computer sci-

[3]We assumed that queries will have nearly 1000 snippets that are downloadable from a search engine. Otherwise, we would not have known whether we were measuring the Kiwi performance or the quality of the queries. This condition could not be completely fulfilled for many collocations, however, and this degraded Kiwi's performance. Here, for QA, this difficulty of obtaining suitable queries is even more strongly emphasized.

ence field, all of whom have been using the Internet for more than 5 years. Each of the students was requested to answer TREC-like questions in Japanese by using either AltaVista or Kiwi. The 32 questions were constructed based on web site usability test guidelines [15]. Examples of the questions are shown in Figure 6. The questions were divided into two groups of 16. 10 students answered the first group by using Kiwi and the second with AltaVista, whereas the other 10 students answered the second group with Kiwi and the first group with AltaVista. Each student was asked to answer the 32 questions by using Kiwi and AltaVista in turn. In using both AltaVista and Kiwi, the students were asked to create queries on their own to answer the questions. They were allowed to refine these queries while searching for the answers. Also, they were allowed to use '/'(comparison) and '+'(And search) with Kiwi and the full functionality of query construction with AltaVista.

The results are shown in Table 3. Each block indicates the time, number of clicks, confidence score, and accuracy for Kiwi and AltaVista, while the columns represent the averages and deviations of the values through 10 subjects. Kiwi outperformed AltaVista in all categories: with Kiwi, users could get more accurate answers faster, with fewer clicks and more confidence. Note that the method of formulating a query, which was totally up to the students in this experiment, remains an issue with Kiwi, as well as with AltaVista. Still, the students could get answers faster with less action with Kiwi. Additionally, the deviations were smaller for Kiwi, meaning that many of the users could use Kiwi in a more stable manner. We guess that this results from Kiwi's totalization of search engine results, so that it helps users think based on the arranged information.

The performance of Kiwi was reduced for questions of the following two types. The first type was a question whose answer was not available on the Web: in this case, the users could not obtain the correct answer. Note that the subjects who used AltaVista had the same problem with this sort of question.

The second type was a question whose query form does not match with its form in Kiwi. Currently, Kiwi only supports one wild card per query. When the subjects faced a question like No. 4 in Fig. 6, however, they all faced the problem of not being able to enter several wild cards. A user-friendly interface thus plays an important role, and we still have to consider this viewpoint in our future work.

7. EVALUATION WITH DIFFERENT ENGINES

7.1 Performance Difference

So far, we have used AltaVista as the search engine for Kiwi. In this section, we examine the differences in Kiwi's results when different search engines are used.

First, we compared the performance difference for collocation. We chose AllTheWeb and Google as the next two engines for evaluation. Note that they use different indexing and ranking strategies from AltaVista, although most of the details of these strategies are not open to the public for commercial reasons. The most important difference, above all, is that Google and AllTheWeb use word-based indexing [2]: for example, a query of "moon*" will not generate candidates such as "moonlight" or "moonshot". This is the same for Japanese and Chinese: Google and AllTheWeb depend on chunkers.

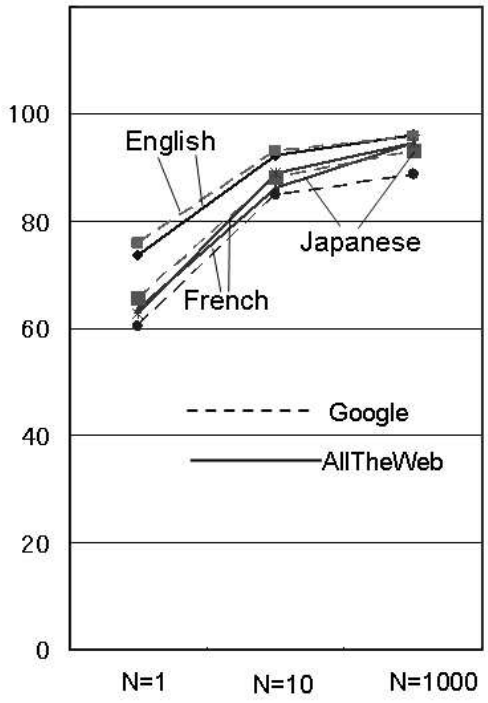

Figure 7: Accuracy of Kiwi using AllTheWeb and Google

The results are shown in Figure 7. As in Figure 4, the horizontal axis indicates different values of N, whereas the vertical axis indicates the accuracy. Each line corresponds to a language (English, Japanese, French) that was tested with one of the search engines (AllTheWeb or Google). The AltaVista results were already shown in Figure 4.

Generally, not much performance difference was observed from the point of view of accuracy, although the search engines adopt very different indexing and ranking strategies. In English, where data is abundant, the performance was especially similar, whereas in Japanese and French, some differences were seen.

Usually, AltaVista outperformed the other two, because of its indexing strategy. This trend was emphasized in Japanese.

7.2 Commonness among Candidates

Next, we verified the *commonness* of the candidates proposed by Kiwi when using different search engines. The results are shown in Figure 8. The horizontal axis indicates different values of N that were verified, whereas the vertical axis indicates the percentage of common candidates. The common candidates were verified between two different engines (i.e., AllTheWeb vs. Google, AltaVista vs.Google, AltaVista vs. AllTheWeb) before being averaged into one score. Each line corresponds to a language.

The commonness rate was the highest in English. Curiously, French then came before Japanese. In English, 80 to 90% of the candidates were the same among any two search engines when the best ranking candidate was verified. This was reduced to 50 to 70% when the top five were examined,

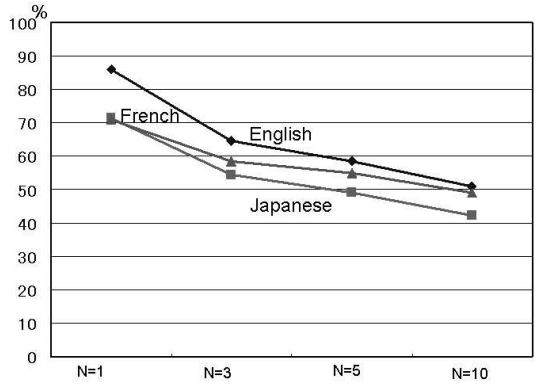

Figure 8: Commonness of candidates among AltaVista, AllTheWeb and Google

because the sets of top n candidates for different engines included different candidates when n was increased[4]

The commonness was the lowest in Japanese when the results were compared between AltaVista and the other two engines. This was due to the different string-based indexing strategy adopted by AltaVista for Japanese.

Still, the concordance of candidates shown here is significant, even though search engines with totally different methodologies were utilized. This commonness was due to the totalization process in Kiwi. This leads to a hypothesis: that if sufficient data is available, the distribution of the aggregated search engine results is independent of the current ranking methodologies. This holds if all search engines handle all documents on the Web and Web data is abundant. Of course, this is not the case, in fact, and all search engines can handle only a portion of the documents on the Web; therefore, this hypothesis does not hold so trivially. Even so, we may observe that the data will trend towards convergence.

Such a hypothesis gives a solid background for what is being researched in NL and IR based on search engines, in the sense that the results obtained using a search engine will be similar even when other search engines are used. Through our work, we think that we have observed a small piece of evidence supporting this hypothesis.

8. RELATED WORKS AND DISCUSSION

Studies on usage consultation using the WWW are rather limited as far as we know. Apart from WebCorp [17], Google Fight [6], and Google Duel [13], Keller et al. [8] argue that high-quality n-grams can be derived from the ostensibly noisy WWW. In the latter case, the target was obtaining adjective-noun, noun-noun, and verb-object bigrams in order to solve the problem of data sparseness in using corpora. The basic philosophy of using the WWW for obtaining usages is a common thread in our work.

In contrast, if we change the viewpoint from usage to QA, there are many works, especially those that put emphasis on the use of the WWW. Among the most recent ones, [10] compared the performance of their MULDER QA system to that of AskJeeves on questions drawn from the TREC-8 question answering track. They found that MULDER's recall is more than a factor of three higher than that of AskJeeves. This study supports the possibility of applying our usage tool for QA tasks.

In exploiting Kiwi's feature of totalization of search results, we might go further and utilize contexts for structuring the resulting usages. Giving a richer context to organize usages might help as much as eliminating noise. This can be done in one of two ways. The first is to adopt classification of documents, in the manner shown by [7] or [9], or by structuring snippets using the technology proposed in [18]. The second way is to organize resulting usages directly by utilizing the user context. The techniques described in [4], where search is performed using the context provided by a user document, may be applicable.

9. CONCLUSION

Kiwi is a web-based usage consultation tool employing search engines. When a user enters a string of words to check their usage, the system sends a query to a search engine to obtain a data related to the string. The result is then statistically analyzed, and the results are displayed.

Our system differs from existing systems in two ways. First, users can enter more flexible queries than with other existing software for related purposes. Second, Kiwi is language independent, so queries can be made in any language if the search engine supports that language. This is achieved by string-based processing of the candidate extraction and ranking. We have formalized the extraction through character branching, and Kiwi ranks candidates based on frequency and length.

According to our evaluation, the missing parts of collocations were provided by the highest-ranked candidate at a rate of 70% to 80% in English, Japanese, and French. When the top ten candidates were considered, nearly 95% of the answers were found. Also, a user evaluation was conducted by requesting subjects to solve TREC-like questions. Kiwi outperformed AltaVista in terms of user performance.

Although search engines are designed for general use, rather than language-specific use, we have found that they can be useful for usage consultation when the results are aggregated. Moreover, Kiwi results with different search engines demonstrated similar distributions of usages in our investigation based on collocations. Users should bear in mind, however, that the results are biased by the amount of available data related to certain topics.

Currently, Kiwi is available to the public at [16].

10. REFERENCES

[1] E. Brill, J. Lin, M. Banko, S. Dumais, and A. Ng. Data-intensive question answering. In *TREC*, pages 393–400, 2001.

[2] T. Calishain and R. Dornfest. *Google Hacks*. O'Reilly & Associates, 2003.

[3] S. Dumais, M. Banko, and E. Brill et al. Web question answering: Is more always better? In *SIGIR*, pages 291–298, 2002.

[4] If engine A ranks results in the order of r_1,r_2,r_3, whereas engine B ranks them as r_2,r_3,r_1, then the overlap increases from 0% to 50% and finally to 100%. Such cases are infrequent, however, and B tends to have candidates ranked as r_1,r_4,r_5.

[4] L. Finkelstein, E. Gabrilovich, Y. Matias, and E. Rivlin. Placing search in context: The concept revisited. In *WWW Conference*, pages 406–414, 2001.

[5] T. Frantzi and S. Ananiadou. Extracting nested collocations. *16th COLING*, pages 41–46, 1996.

[6] GoogleFight. Google fight : Make a fight with googlefight, 2000. Available at `http://www.googlefight.com/`.

[7] T. Haveliwala, A. Gionis, D. Klein, and P. Indyk. Evaluating strategies for similarity search on the web. In *WWW Conference*, pages 432–442, 2002.

[8] F. Keller, M. Lapata, and O. Ourioupina. Using the web to overcome data sparseness. In *EMNLP*, pages 230–237, 2002.

[9] I. Ko, K. Yao, and R. Neches. Dynamic coordination of information management services for processing dynamic web content. In *WWW Conference*, pages 355–365, 2002.

[10] C. Kwok, O. Etzioni, and D. Weld. Scaling question answering to the web. In *WWW Conference*, pages 150–161, 2001.

[11] U. Manber and G. Myers. Suffix arrays: a new method for on-line string searches. *SIAM Journal of Computing*, 1993.

[12] H. Nakagawa and T. Mori. Automatic term recognition based on statistics of compound nouns and their components. *Terminology*, 9(2):201–219, 2003.

[13] G. Peters. Geoff's GoogleDuel!, 2002. Available at `http://www.sfu.ca/ gpeters/cgi-bin/pear/`.

[14] G. Reach. Online language populations, 2004. Available at `http://www.glreach.com/globstats`.

[15] M. Spool. *Web Site Usability: A Designer's Guide*. Morgan Kaufmann Publishers, 1998.

[16] K. Tanaka-Ishii and H. Nakagawa. Kiwi site, 2004. `http://www.kiwi.r.dl.itc.u-tokyo.ac.jp/`.

[17] Webcorp. Webcorp home page, 1999. Available at `http://www.webcorp.org.uk/`.

[18] O. Zamir and O. Etzioni. Grouper: A dynamic clustering interface to web search results. In *WWW Conference*, pages 123–131, 1999.