Knowledge Based (KB) Phrase Recognition

Syed Raza
Talladega College
sqraza@talladega.edu

Abstract

The study concerns automating one's cognitive ability to recognize that a text contains a word that does not fit its context and is not flagged as an error. How the misfit is discovered by a human reader is discussed and focus is narrowed to the case of the context being a commonly used phrase or saying. A model that incorporates an efficient strategy for identifying the presence of the phrase in a text stream is developed.

Introduction

"Knowledge Based Phrase Recognition," refers to the storing of commonly used phrases in order to use such explicit knowledge to discover when an incorrect word is in a recognized instance of a phrase. Much of the study concerns the issues and details involved in recognizing a phrase that is only partially correct. Analyses of knowledge management (KM) have identified two key transitions: converting tacit knowledge into explicit knowledge and vice versa [1]. This study focuses on a tacit skill that many use: the ability to notice in a text when a wrong word is used. For example, a paragraph in a student paper might be about the severity of a problem but the topic is written as the "severance" of a problem. It is worth trying to automate this skill as a text monitoring tool, using fairly simple tactics.

Unfortunately, spell-checkers do not find a wrong word when it is well spelled, but the proposed Wrong -Word Finder (WWF) does resemble a spell-checker. Both tools scan text to test the correctness of words. Both require that the scanning system has a list of acceptable patterns. But while in the spelling case a stored pattern is a word and its parts (letters) are verified, in the WWF case a stored part is a pattern of words and its parts (words) are verified. For spelling, word correctness depends on the word's internal structure; in contrast, at the phrase (WWF) level, word correctness depends on the word's fit to its external context. In this paper these external contexts that might be recognized by the WWF are called "phrases", and the list of acceptable patterns is considered a "phrase dictionary" (PD).

The WWF has a problem that is not encountered in spell-checkers. In the text being scanned, the WWF does not have explicit delimiters for a pattern in a text that might also be in the PD. Word boundaries are explicitly well marked; boundaries of phrases are not. A key task in this project is to determine how one might make the transition from a string of words being scanned to locating a relevant entry in the PD.

The WWF must have a way to discover in a text a partially correct phrase that is similar to a PD entry, even though one or more of the words will not match the PD entry. Almost any word in the sentence might be the beginning of a PD entry. Moreover, the beginning word might be a word that is wrong in a phrase that otherwise matches a PD entry. Unlike a spell-checker, which has a well-define target *before* consulting its dictionary, the WWF *uses* the PD both to determine the boundaries of the target phrase and to check the correctness of the text words against a PD entry.

The proposed WWF will also operate without a great deal of linguistic processing. It can be argued that many familiar phrases, especially idioms, are connected only to a global meanings and appropriate contexts for using them. One might use a phrase (such as “the whole 9 yards” or “lock, stock and barrel”) without really understanding its original, detailed meaning. Using grammar to find phrases in the sentence has its drawbacks. If one learns a whole phrase by rote, there is no guarantee that the resulting PD entry is grammatically correct. Moreover, it is argued that parsing the sentence produces too many phrases to check and that a much simpler way to find phrases exists.

Related Conceptual Areas

Related work can be found in human information processing, computational linguistics, and knowledge software design. The principle contributions from human information processing and related fields are: schema, context determination of meaning, and memory chunking [2], [3], [4]. They represent a consistent view of human cognition. A *schema* is an expectation stored or reconstructed by the human brain and is essential to understanding how the brain recognizes anything at all. In a sense, we have mental maps, scripts, and models that both account for input and predict future input [5], [6], [7]. For the purposes of WWF, a schema captures the idea that stored patterns (phrases) are used to recognize input.

Contextual determination of meaning is an even earlier psychological concept. It was especially stressed by Gestalt psychologists [8], but it was recognized immediately in cognitive psychology and related cognitive sciences [9]. The meaning of a target is determined in part by its context. For example, if one were to sniff strongly just before uttering the well formed English sentence, “I dot a code in my node”, such a context might well cause the listener to hear “I’ve got a cold in my nose.”

Since 1956, the idea *chunking* has been an important concept [10], although initially it referred to a way around a short term memory capacity that allowed only a handful of units to be retained simultaneously, but chunking was a way around the limit by making the units larger chunks. For example, 1491625364 exceeds STM capacity as digits, but not if partitioned into chunks: 1491, 625, 364. Computational linguists are also interested in chunking as a form of segmenting a sentence before attempting to parse it [11].

.

A computational linguistics distinction that can be made in the WWF project is that information from the input (e.g., a word) can be used (in *bottom up* fashion) to retrieve a phrase and then the rest of the checking is done (in *top down* fashion) from the phrase toward the words. Another distinction to be considered is *context sensitive* processing versus *context free* processing. The most inclusive strategy for finding a wrong word would be context sensitive, for the opportunities to select an inappropriate word are open ended and depend on the meaning of the word's context But when the context is a common phrase stored in the memory of the WWF, the word is evaluated in the course of a context-free *phrase* checking, a much simpler and doable task.

Two strategic ideas in knowledge software design are so important to the project that they are given special emphasis: context based processing and conversions between tacit and explicit knowledge. As in the human case, *context* is a major factor in such tasks as correcting misspelling [12], [13], [14].

Tacit knowledge today still causes lively debate in the context of knowledge management [15], but it was popularize as a concept in human problem solving [16]. Tacit knowledge resides within the human and is what the human understands. A message may contain information that can be related to tacit knowledge; but two humans may relate such explicit, public representations to very different tacit understandings. Politicians are well aware of such a condition. The sources cited above stress that conversion from tacit to explicit (symbolizing incompletely tacit understanding) and converting explicit back to tacit understanding (open to ambiguity and bias) are major problems in knowledge creation. The WWF project is an attempt to create an automated explicit alternative to what humans can do tacitly.

Considering How Humans Notice Wrong Words

This project is influenced by ways human monitoring and error detection occur. For instance, in a history of philosophy text the phrase “imminent in nature” occurred, but memory of a similar sounding phrase from philosophy competed in the reader's mind: philosophers more frequently use the phrase “immanent in nature”. Further investigation of the paragraph in the text confirmed the suspicion that the wrong word was used. There are many pairs of words that are confused, such as: further-farther, empirical-imperial, elicit-illicit, principle-principal. Spelling and grammar checkers fail to notice such cases, but often human readers can notice that a word choice is wrong.

Apparently, one often anticipates from the context what word should be present and notices it is not there. After hearing phrase in several detective dramas, a young man asked his mother, “What is a tooth-comb?” His mother discovered the relevant context and corrected him, “Oh, you go over the evidence with a comb that has fine teeth, not a fine comb to use on teeth.”

A phrase is sometimes acquired with only an understanding of its meaning as whole. Its general meaning and when to use it are remembered but not necessarily any deep analysis of what its components mean. While fishermen would notice the detailed meaning and origin of “hook, line and sinker,” probably few people would be able to describe the detailed meaning of “lock, stock and barrel” or “between the devil and the deep blue sea.”

In general, not all the words in a sentence are equally significant. Some words set the possible topics to which the phrase points. For example, perhaps in “Now is the time for all good men to come to the aid of the party” only “now”, “time”, “aid”, “party” are main words. Each of these topic-inducing parts of the stored phrase can potentially remind one of the stored phrases. While "imminent" would not by itself remind one of "immanent in nature", "nature" might access "immanent in nature." (But in this case, the close superficial resemblance of "imminent" and "immanent" probably also plays a role.) A comparison of what comes from memory and what comes from reading the sentence reveals that a wrong word has been found.

Approaching Automated Detection

A natural solution to the "wrong word" problem seems to be the use of context to anticipate what the right word should be. The processes of anticipating and using a potential context of a target relates to well researched human processes, such as: the use of perceived context or situation to determine meaning [17], [18] and the existence of stored patterns or schema [5], [6] ,[19].

One strategy is to use the strength of semantic associations to develop an automated system that decides the appropriateness of a word in a context; for example, the system expects "principal" and not "principle" when "school" is nearby in the text [20]. This may well be a major way a human corrects wrong word usage; but in the present study another idea is pursued: that the human recognizes a sequence of words as a familiar pattern, one that is close to a pattern stored as PD entry.

Other pending strategic issues include:

- Should the study effort be directed towards accomplishing the task in the same way that it is done by humans or should non-human ways of processing information also be considered?
- How much linguistic knowledge is to be used? When may semantic associations be ignored?
- Should there be an effort to identify phrase boundaries within the sentence *before* searching for a corresponding item stored in a memory?

PEDS: A KB Phrase Recognizing WWF

The remainder of the paper focuses on designing a Phrase Error Detection System (PEDS) as an example of a knowledge-based phrase-recognizing wrong word finder. The overall approach involves applying standard software engineering to design a WWF and address the questions and issues raised above.

PEDS uses three data sources: an input sentence, a phrase dictionary (PD) and a structured dictionary of keys (KD) that will enable a scan of the sentence to access relevant entries in the PD. The type of information stored in KD will be an issue. An entry in KD must associate both to part of the sentence and to relevant entries in the PD. Two crucial and related issues are: how to relate a fragment of the text to a PD entry and what shall be the role of the KD information.

Top-Down versus Bottom-Up Approach?

Two issues concerning sentence processing are:

- How much pre-processing of the sentence is needed? The top-down KD approach requires that the sentence be pre-processed according to the same rules that created the KD key. In the bottom-up KD approach, no pre-processing is needed; the scanner will traverse the sentence word by word.
- When only a part of a sentence is being tested for relevance, how to ensure that the segment boundaries do not discard a part of a meaningful phrase?

The second of the above issues (that is, what fragments of a sentence shall be compared with a PD entry) also concerns top-down versus bottom-up strategies. The top-down strategy would have such information in the KD entry. That is, by some pre-set rules every PD entry would be condensed from a word phrase into a sequence of letters and symbols. For example, the first letter of every word in the phrase could be stored in and KD entry. One variation of this is that only the first letters of main words be stored and a wildcard symbol, such as "*", would be represented gaps between the main words. Thus, in condensed form, the overall structure of the PD entry is condensed into an entry in KD. Before the input sentence could be processed, a condensed version of it would have to be made using the same rules as used for creating the KD key.

While the top-down strategy preserves the overall structural (sequence of initials) information in KD, there is a serious drawback. To work, every entry in KD must be considered in the sentence scan. This would happen by using a function that took each string in KD together with a portion of the condensed sentence and had as output a measure of what proportion of the characters matched. If enough of a match occurs, the corresponding PD is retrieved and a word by word matching is tried. This condensation strategy creates overhead that may save time and space for large dictionaries and input texts, but the comparison goes in the wrong direction. The strategy runs through a very large set of possibilities in KD in search for a hit in the condensed sentence.

In an input text there are two boundaries that are well marked: sentence boundaries and word boundaries. By segmenting the input text into sentences one can be certain that no phrase is truncated by the segmentation. The relevant phrase boundaries are not immediately needed. The bottom-up approach selects in turn each word in the sentence and checks in KD for its match. No condensing occurs. More importantly, the KD entries can be organized when loaded so that an efficient search of KD can occur. For example, if all the KD entries were in alphabetical order, a binary search for the word could occur. Ordering of KD only needs to occur during the learning period, when the PD and KD are loaded.

The presence of a word as a KD entry means that word plays a significant role in at least one PD entry. For example, there would surely be a KD entry for the word "love", and that entry would point to every entry in PD in which "love" is a main word in the phrase. In other words, the KD information is that a word is used significantly in one or more entries in the PD. Only those PD entries linked to the KD would be retrieved. Moreover, the PD entry contains the information by which the sentence phrase boundaries can be found. In sum, WWF uses the sentence word to find through KD a small set of PD entries that may be relevant; WWF then uses the short list of potentially relevant PD entries to find a PD that at least partially matches. It would be up to the knowledge worker loading PD and KD to abstract a sequence of words from the PD entry for use in adding to the KD.

Consider the phrase “a knight in shining armor” as an entry in PD. Assume the knowledge worker decides that the three long words are all significant. KD links would be added for all three words. If any of them, such as “shining”, already were in KD due to another PD entry, then just an address to the new PD entry would be added. Otherwise, a new node in KD would be created and pointed to “a knight in shining armor” in PD. Because there were three significant words in the PD, there will be three links added in KD.

Suppose the text sentence had the phrase “a night in shining armor.” (google.com had over 90,000 hits for exactly that phrase.). The sentence word “night” would trigger the KD entry “night”, which would provide those PD entries with significant usage of “night”, but none would match a sequence in the sentence. Presumably, small words such as “a”, “in”, “the”, would not be put in KD, so a binary search for them would be quickly resolved. But “shining” would be found in KD and by straightforward brief searches of the sentences it would be discovered that “night in shining armor” is a partial match to “knight in shining armor”.

In sum, a key issue concerning the information in the KD is whether the entries should reflect a top-down versus a bottom-up approach. A top-down approach would preserve some of the structural information from the PD entry, in condensed form. In the bottom-up case a KD entry would be a word that could be related both to the sentence and a relevant PD entry. In both cases, the KD entry would point to one or more PD entries.

These decisions about efficiency have been made:

- One efficient strategy is the use of a word from the input sentence to find (bottom-up) a phrase in the PD and then use the phrase (top-down) to check for wrong words. To be able to discover the end of a phrase in the sentence due to having the relevant PD entry information also improves efficiency by determining where to find the next word in the sentence.
- Using a binary search of KD is also important.
- In order to adjust to normal variations in wording, it will be efficient to do some pre-processing of target words, so that changes in such as tense or number will not prevent finding keys in KD.
- Efficiency will be relevant for the process of comparing the PD entry with the text segment.

The Phrase Dictionary

Several issues concern the contents of the PD, such as:

- How much of a phrase should be stored? For example, which should be stored: “He bought it lock, stock and barrel” or “lock, stock and barrel”?
- What will be the minimum number of main words? Here the problem is that allowing some words to be wrong and still be able to find the correct PD entry means that the PD entry must have other words that can be used to find the PD entry.
- Should one selection criterion be the likelihood that a phrase will be misquoted? For example, the common lack of understanding about the origin of “whole 9 yards” makes it

more likely it will be transformed into “whole 10 yards”. And maybe “level playing field” could become “level plain field” or “level plane field.”

- What allowances should be made for normal variation in wording, for such factors as number and tense? Should the PD entries be somewhat abstracted in order for them to work with text that has been adjusted for tense and number during pre-processing?
- What types of phrases might be stored? Some types include familiar quotes, idioms, clichés, phrases that are often misquoted. There are several common types of phrases that might be included in a PD. There are famous quotes and mottos, such as “Honesty is the best policy” and “Ask not what your country can do for you, but what you can do for your country”. Common idioms include, “between a rock and a hard place” and “down in the dumps,” clichés and analogies such as “Y’all come back some time, y ’hear?” and “It went over like a lead balloon.” Then there are the short frequently used transitions, such as “in the meantime,” “that is to say”, and “by the same token.”

As a practical matter, there are too many candidates like those above. Yet almost any phrase can be misunderstood. A suggestion is that at least the first entries in the PD ought to be those which the knowledge worker thinks have a high chance of containing a wrong word. For example, assuming one does not know the actual origin of the “whole nine yards” and presumes it to be related to football. It is quite plausible that a fan would believe the meaning is still preserved in “whole ten yards”. How many main words should a PD entry have? Here again, the discretion of the knowledge worker is needed. Enough main words should be stored in KD so that if some words are incorrect in the sentence, there are still some left by which to identify the PD entry. Too many words stored in the KD will slow down the performance and reduce efficiency.

The PD does not have to be organized, since access to it from KD can be by hashed addresses. Both the full phrase and the sequence of main words should be in a PD entry. A typical PD entry might be an ordered list such as: “‘A bird in the hand is worth two in the bush,’ bird, hand, bush.”

PEDS Procedures

The following general steps are taken in the Phrase Error Detection System and Figure 1 represents the overall design of PEDS.

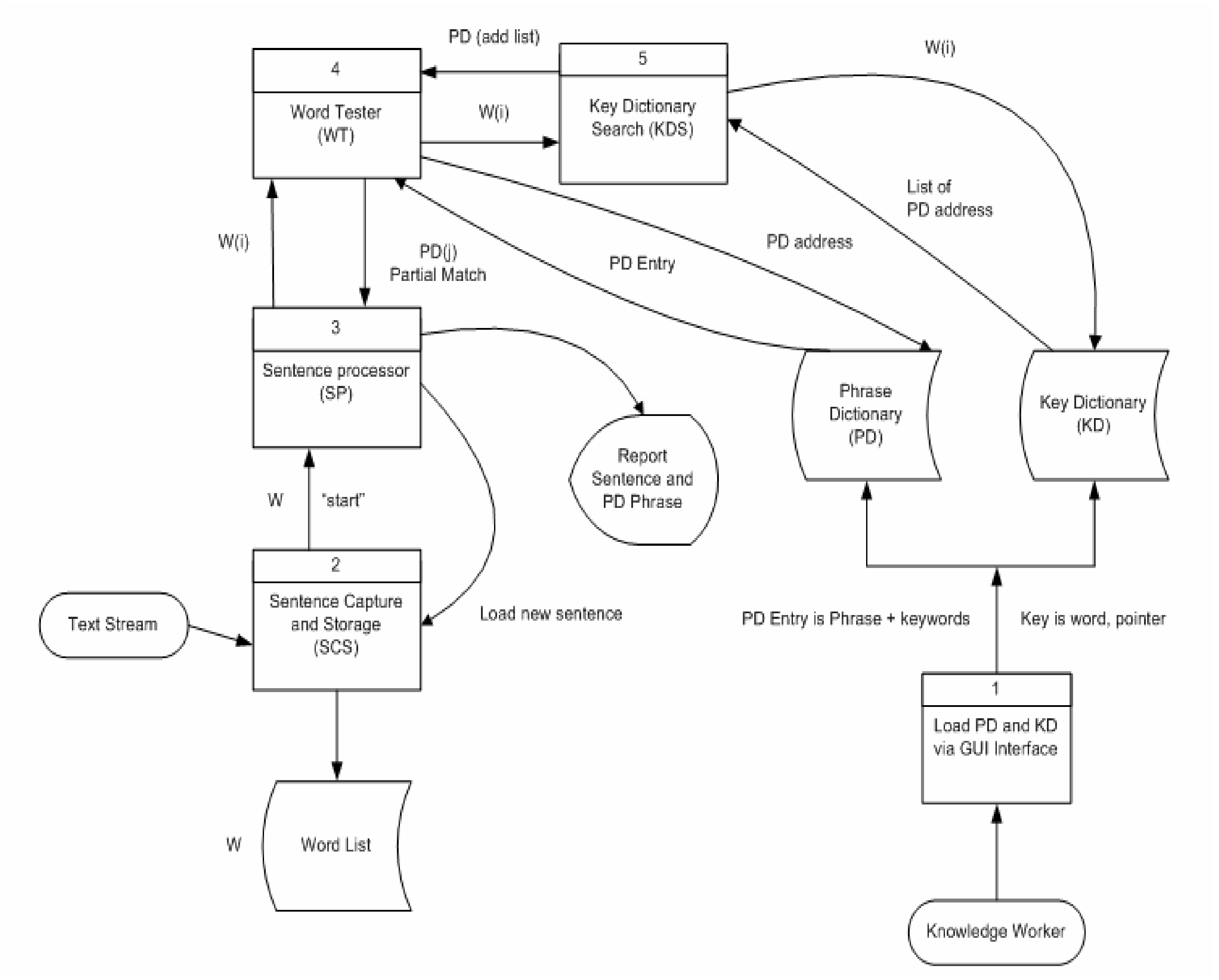

Figure 1: The Phrase Error Detection System (PEDS) Architecture

Figure 1 indicates that a human knowledge worker (KW) has to decide when a phrase is to be put in PD and which words in the phrase are to be entered into KD as keys to the phrase. The following other processes then occur:

1. The Loader receives information from a KW and loads it appropriately into the PD and KD, including KD pointers to PD entry. When the loader is activated at the user interface, it makes a separate KD entry for each new key and a pointer to the phrase being entered in the PD. If the key is already in KD for another phrase, an additional pointer is added to the existing key. The PD entry is structured as a phrase, followed by a list of words that would be checked when compared with a text segment. The KD structure is a word, followed by a list of pointers to PD entries for which the word is a key

2. In the Sentence Capture and Storage Component (SCS) each sentence is extracted from input text stream by use of standard punctuation and stored as a word list (W), which is also passed to the next component, SP.
3. The Sentence Processor (SP) picks the next word, W(i), from word list in W and sends W(i) to the next component, WT. SP will receive back either a partially matching phrase or a null ("not found") signal. If a phrase returns, it is reported it out along with corresponding sentence context.
4. The Word Tester (WT) receives the word from SP and queries via KDS for its being in KD. If it is in KD, WT receives from KDS a list of addresses of phrases in PD. WT then retrieves the phrases and passes them back to SP. See Module 4 algorithm below.
5. The KD Searcher (KDS) is given a word from WT and returns pointers to phrases in PD if word matches an entry in KD. See Module 5 algorithm below.

The following are sample algorithms:

Module 4: Word Tester (WT).

```
(Note: module 3 passes target word W(i) to module 4)
    Pass W(i) to module 5
    Accept address list returning from module 5
    IF list is empty
        return to module 3 with no match signal
    ELSE start using list to retrieve PD entry
        For each address on list DO Loop
           Get entry from PD
                (entry is phrase and sequence of main words)
           Use list of main words to search for corresponding words on
                sentence word list working out from the target (W(i)) in both
                directions.
         IF enough words match
                Return to module 3 with the phrase
         END Loop
          Return to module 3 with a no match signal.
```

Module 5: Key Dictionary Search (KDS)

```
    INPUT:      W(i)  from wordlist
    RETURN:     A list of PD addresses possibility an empty list
        Binary Search alphabetically order KD list (Concordance) for W(i)
       (A KD entry has a word + a list of PD addresses)
    IF  W(i) IS FOUND
        Return list of PD address
    ElSE
        Return with Empty list
```

A sample interface is used in the Loader. The PEDS is implemented as a stand alone system, where the client enter the phrase (as a string) and the keys to be stored in the Key structure (KD) and Phrase Dictionary (PD), which accept the both the string buy pressing the button "Load". Figure 2, shows the screenshot of the Interface.

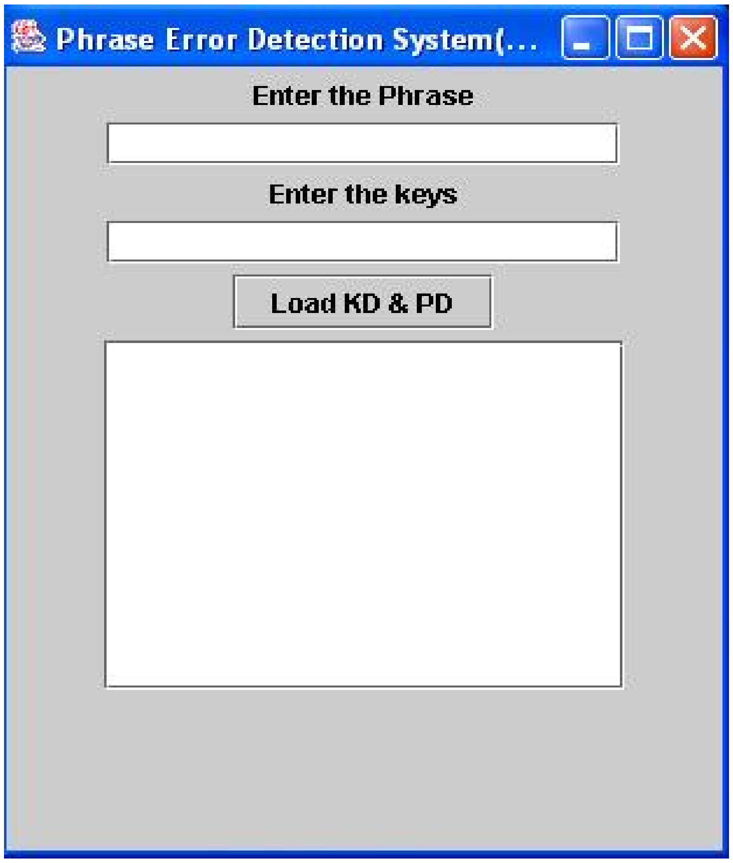

Figure 2: A screenshot of the Interface, loads KD and PD

Conclusion

It is encouraging that many functions of interest to this project already exist, but have not been combined to achieve the following functionality: that a context can be recognized efficiently and used to detect an error within the context. Spell-checkers find words and verify them against a lexicon. Grammar-checkers use parser generated context to verify grammatical soundness. The context required to identify errors that are neither spelling or grammatical errors is too open-ended for using the generative rules of a grammar and not well marked as in the case of spelling checkers; but nonetheless the paper has explored a way to find one of a common class of contexts by using its parts as keys to recognizing the context.

The approach affords substantial generality, wherever there is chance to expect from previous learning that a target should fit in a particular context. Maybe someday the information about a fragment of DNA will allow a researcher to discover that the expected context of the fragment reveals that the input DNA has a flaw.

Future developments of this approach can be in the direction of anticipating phrases in real time, just as happens currently when one begins to type in the day's date: an option appears that offers to complete the date without further keystrokes. There could be automated suggestions for more accurate phrasing, just as alternatives spelling replacements are now offered. Maybe a text writer will seldom have to finish a cliché that has been started. (And maybe the very recognition that it is a cliché based on the automated feedback will discourage the over-usage of clichés.).

There are a number of examples of filling-in tools; dates and often used names are offered while one is in the midst of typing them. In Microsoft Word, there is a memory for recently created options to paste; but such items are specific to the session. Moreover, it is apparent that search engines can rapidly find even obscure targets; it is crucial to WWF that identification of correct phrases in context occur in time to be useful.

As suggested earlier as an efficiency tactic, WWF can use abstract versions of key words. This would be implemented as word stems instead of whole words. It would require dropping well-known inflectional suffixes, such as –ing, and –ed in a target word from the sentence and then still doing a binary search on the KD. The KD itself could be a concordance of word stems. Alternatively, the common function of comparing one string with another in search of the one being a substring of the other can be used on KD. This extension does introduce some linguistic knowledge- just enough to increase flexibility and efficiency.

References

[1] Nemati, H. R., Steiger, D. M., Lyer, L.S. & Herschel, R. T. Knowledge warehouse: an architectural integration of knowledge management, design support, artificial intelligence and data warehousing, *Elsevier Science B.V., 2002*, 33,143-161.

[2] Neisser, U. & Hyman, I. *Memory Observed: Remembering in Natural Contexts*. (2nd ed.). London: Worth Publishing Ltd., 1999.

[3] Newell, A., Simon, H. A. *Human Problem Solving.* Englewood Cliffs, NJ: Prentice Hall, 1972.

[4] Sanford, A.J. *Cognition and Cognitive Psychology*.(1st ed.). New York, NY: Basic Books Publisher, 1986.

[5] Anderson, J. R. Concepts, propositions, and schemata: what are the cognitive units? *Nebraska Symposium on Motivation, 1980*, 28, 121-62.

[6] Neisser, U. *Cognition and Reality: Principles and Implications of Cognitive Psychology*. San Francisco: W.H. Freeman, 1976.

[7] Neisser, U. Stories, selves, and schemata: A review of ecological findings. In M.A. Conway, S.E. Gathercole & C. Cornoldi (Eds.), *Theories of Memory II*. 171-186. Hove, UK.:Psychology Press, 1998.

[8] Kohler, W. *The Task of Gestalt Psychology*, Princeton, NJ: Princeton University Press, 1969.

[9] Gillespie, D. *The Mind's We: Contextualism in Cognitive Psychology*. Carbondale, IL: Southern Illinois University Press, 1992.

[10] Miller, G. A. The magical number seven plus or minus two: some limits on our capacity for processing information. *Psychology Review*, 63, 1956.

[11] Abney, S. Parsing by Chunks. In Berwick, Abney & Tenny, eds. *Principle-Based Parsing*, (pp. 257-278). Dordrecht: Kluwer Academic Publishers, 1991.

[12] Al-Mubaid, H. Context-Based Word Prediction and Classification. *In Proceedings of 18th Intl Conf on Computers and Their Applications(CATA)*, Hawaii, 2003.

[13] Al-Mubaid, H. & Truemper, K. Learning to Find Context-Based Spelling Errors, in Data Mining and Knowledge Discovery Approaches Based on Rule Induction Techniques, *Kluwer Academic Publishers, 2004.*

[14] Al-Mubaid, A. & Chen, P. Context-based similar words detection and its application in specified search engine. *ACM Press, 2005*, 260-262.

[15] Hildreth, P. M. & Kimble, C. The duality of knowledge, *Information Research*, 2002, 8(1).

[16] Polanyi, M. *The Tacit Dimension*, London: Routlege and Keoan Paul, 1967.

[17] Aronson, E. *The Social Animal* (9^{th} ed.). London: Worth Publishing Ltd. UK, 2003

[18] Kukich, K. Techniques for automatically correcting words in text. *ACM Computing Surveys*, 24(4), 1992.

[19] Bartlett, F. C. *Remembering*. Cambridge: Cambridge University Press, 1932.

[20] Mangu, L. & Brill, E. Automatic rule acquisition for spelling correction. *Proceedings of the Fourteen International Conferences on Machine Learning*, 1997, 187-194.

Biography

SYED RAZA is Associate Professor of Computer Science and Chair of the Department of Mathematics and Computer Science. He earned his B.Sc. (Science, 1993), M.S. (Computer Science, 1998) degrees from University of the Punjab. He earned his M.S (Computer Science, 2004), Ph.D., ABD (Computer Information System, 2006) from Nova Southeastern University, FL. He teaches a variety of courses to undergraduates. His research is in RFID, knowledge management and data analysis techniques such as data mining and its application.