A Natural Language Model of Computing with Words in Web Pages

Zheng Ze-yu[1], Zhang Ping[2]

[1]School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, Hubei, 430074, P.R. China
[2]School of Foreign Languages, Zhongnan University of Economics and Law, Wuhan, Hubei 430064, P.R. China, seekinghome@126.com

Abstract: This paper is to deal with probabilistic finite automata (PFAs) by inputting strings of words (probability distributions) in web pages of Internet, and investigates the development of a natural language (NL) interface for mixed initiative dialogues within a constrained domain and demonstrates the applicability. Specifically, it is verified that PFAs computing strings of words can be implemented by means of calculating strings of symbols and a characterization of PFAs. On the other hand, a natural language model system in Internet is discussed. The system is written entirely in a purely functional language, which results in a surprisingly small and simple program. Finally, the main results obtained are summarized, and a number of related issues for further study are raised.

Keywords: Natural language processing, Functional programming, Web pages, Probabilistic finite automata

1 Introduction

With the deregulation of the Internet and the development of ever more complex and sophisticated systems such as enterprise resource planning, user relationship management systems, and operating systems it was inevitable that a parallel requirement for more sophisticated online user assistance systems would follow.

Computing in the traditional sense involves inputs with strings of numbers and symbols rather than *words*, and even in those nontraditional models of computation such as molecular, stochastic, analog, and quantum computing, the inputs still are strings of numbers and symbols instead of *words*. In this paper, words mean probability distributions over input alphabet, and are different from the words in classical formal languages and automata theory that represent strings of input symbols. The major technical contributions of this paper are twofold. On one hand, we study PFAs with the inputs to be strings of words that are probability distributions over input alphabets[1]. On the other hand, we do research on the natural language system which contains the natural language subsystem that comprises a parser that interprets natural language inputs to semantic information and the knowledge-based subsystem that selects the most appropriate script for a guided dialogue to the user with semantic tags extracted by the natural language subsystem.

In the paper, we deal with PFAs with input strings of words. Specifically, we can derive the probability for PFAs computing strings of words by means of calculating these probabilities of PFAs recognizing strings of symbols (Theorem 1), and thus shows that PFAs for inputting strings of words can be transferred to PFAs computing the convention all strings of symbols, at the expense of more steps of computation, which may be called computation tractability.

This paper builds upon the previous research to present a new two-part computational natural language processing (NLP) system based upon the use of an executable set of functional equations and through this notation demonstrates its applicability to the creation of knowledge-based online help systems. A prototype solution to the UNIX help assistant problem is presented, which is felt to be a suitable domain through which the operational aspects of the approach could be tested as its solution space is formally defined, yet facilitates mixed initiative dialogues.

The NLP subsystem accepts as input a BNF (Backus-Naur Form) description of the language[2]. This approach has the advantage that the language module can be replaced or upgraded by any user who understands formal grammars without requiring any programming. The approach taken here is to produce all possible parses of the input query. In the relatively restricted domain of help systems, inputs are not large: queries tend not to be multi-paragraph compositions. So although ambiguities may still produce more than one possible parse, the sometimes exponential explosion of possibilities is not a debilitating problem.

The BNF accepted by the parser is extended with simple semantic tags that essentially say 'if a successful parse comes through here, make a note of xxxx'. The output from the parser is not just a list of possible parse trees, but also a list of possible tag sets. Each tag set contains the semantic tags that are encountered in an ultimately successful parse of the input. For example, in the famous example 'fruit flies like a banana', the word 'flies' can be either a verb or a noun. In the extended BNF, where 'flies' is listed as a possible verb, the tag 'action-travel' could be specified; where it is listed as a possible noun, the tag 'actor-insect' could appear. With a similar treatment for 'fruit', 'like', and 'banana', the parser would produce two possible tag sets:

i. actor-insect, actor-enjoy, object-food ;
ii. actor-food, action-travel, manner-food

The simplified, linear representation of meaning would clearly not be sufficient for a full natural language understanding system, but in the restricted domain of online help systems with its much smaller expected inputs, it provides an appropriate level of detail for further analysis.

The script selection process is a variation of the approach advocated by Hobbs (1995) [2] for the creation of generic information extraction systems and extends of Plant's, as Fig1 [3] work on knowledge-based help systems[4].

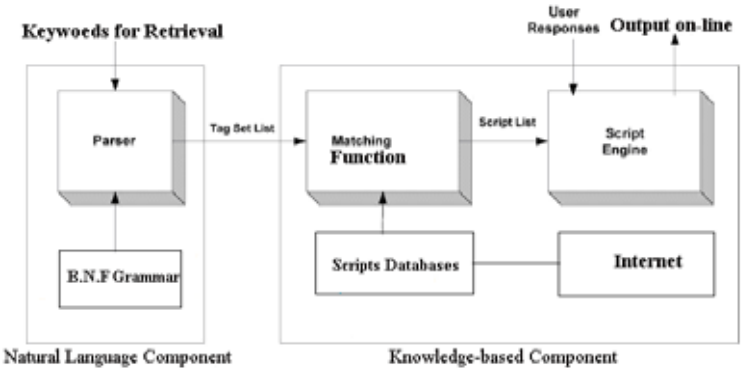

Fig. 1. Components of the mixed initiative dialogue system in Internet

2 Language for Research in Web Pages

A major issue to decide upon is how to represent the BNF syntax as a data structure. However, a pre-requisite factor on the nature and form of the data structure is the language with which the whole system is to be implemented. Several languages are suggested as follows.

The functional language AFL is chosen for many reasons; however the main advantage this class of language has over traditional procedural languages is that it effectively reduces the amount of work the programmer has to perform. It does this in two ways: first, by handling allocation of storage. Second, which is more important, is that the system assumes all responsibility for the evaluation order of the functions, removing the problems associated with structuring the program in order to obtain the desired sequence of evaluation.

The use of infinite data structures is also of significant advantage in that they are ideal for processing the input and output of a parsing program that can be regarded as infinite streams of information. This approach to input/output, by-passes the problem of explicit 'reads' and having to decide upon the

sequencing of all events prior to processing, also by programming in terms of infinite I/O streams this allows program modules to be combined easily [5][6].

Pattern matching is a feature of functional programming languages that is extremely useful in the domain of NLP. These techniques allow complex conditions to be expressed very simply, especially when the functions have many arguments, reducing the need to use guards. The functional approach also allows the use of higher order functions and a recursive equation style of programming.

2.1 Functional features of BNF

Having decided upon the BNF form of grammar and the use of the functional language AFL with which to ultimately implement the parser, the next stage is to devise a representation of the BNF in AFL.

There are four components making up the BNF: (i) Terminal symbols, e.g. words like 'dog', (ii) Non terminal symbols, e.g., noun. these being the name of structural units and denoted by enclosure within angle-brackets, (iii) The disjunction of two or more components, the 'or' being represented by the 'l' symbol, and (iv) The conjunction of two or more components with the 'and' being represented by juxtaposition. We extend this with a fifth component denoting a semantic tag.

For example,

<s>::=<noun>|<verb>

means <s > is either a <noun> or a <verb>, whereas

<s>::=<noun> <verb>

means <s> must consist of a <noun> followed by a <verb>.

It was decided to represent these components in terms of lists and by defining some reserved words to be recognized by the parser.

Translating BNF to AFL, terminal symbols remain in a similar format; the word dog becomes the string "dog". Non-terminal symbols now become single element lists, e.g., noun. would become ["noun"]. The reserved words 'or' and 'seq' were then introduced, allowing the conjunction and disjunction of symbols. For example,

<verb>|<noun>

is represented as ["or",["verb"],["noun"]], and the sequence

<verb> <noun>

by ["seq",["verb"],["noun"]]

Semantic tags are introduced by the reserved word 'tag' .For example:

["tag", ["seq", ["verb"], ["noun"]],"imperative"]

means that ["seq", ["verb"], ["noun"]] should be parsed and a semantic tag added noting the fact that this sentence is 'imperative'.

Whole BNF descriptions are collections of named rules. In AFL BNF is represented as a structured list, the first item in each sublist is the name of the rule and the rest of each list is the definition.

<s>::=<noun> <verb> <noun>::="cat"|"dog"

would be represented by the definition:

BNF==[["s",["seq",["noun"],["verb"]]],["noun",["or", "cat", "dog"]]]

Having devised a representation in which to express these components we are able to write equations for any set of BNF definitions. And the sentence 'fruit flies like a banana' can be expressed as a list of terminal symbols: Sentence=["fruit", "flies", "like", "a", "banana"] The online version of the volume will be available in LNCS Online. Members of institutes subscribing to the Lecture Notes in Computer Science series have access to all the pdfs of all the online publications. Non-subscribers can only read as far as the abstracts. If they try to go beyond this point, they are automatically asked, whether they would like to order the pdf, and are given instructions as to how to do so.

2.2 the Knowledge-Based Processor

Having constructed the parser and a functional specification of the BNF grammar, the next stage is to make the system respond to the user queries with meaningful answers on net. This has been achieved through the use of a knowledge base used as its representational structure an association list.

The association list connects pre-constructed scripts and informative texts to patterns or templates for semantic tag lists. For example, a user wanting to know how to print a file may type 'I want to print a file', which would be reduced by the parser to the tag set [["action", "print"], ["object", "datafile"]]. Information about how to print files may have as its index in the association list [["action", "print"], ["object", "*"]], indicating that it would be a relevant response to a question about printing something. All of the possible tag sets are compared against all of the possible association list indexes, and a score is computed for closeness of match. The entry with the highest score is selected for presentation to the user as the answer. In a more sophisticated version, it would of course be possible to offer the user a 'menu' of the highest scoring entries to select from. This form of knowledge base provides a means of outputting textual answers to direct questions. For example, in the Unix help system, a user may input: I want to print the budget Then, the highest score may be 100 and the answer given by the computer will focus on the command 'lpr' and its use.

A one-sided dialogue of this type could produce responses no better than the existing Unix help systems; one word of input does not provide enough information for anything better. An improvement on this basis, is the provision of a 'mixed initiative' dialogue capability to provide the user with a more stimulating interaction, allowing far more probing questions to be asked and more meaningful answers given[1,6,8].

3 A Mixed Initiative Dialogue

In order to build a mixed initiative framework it is necessary to extend the knowledge base. However, it is felt that the specialized knowledge required for a dialogue should be separated from the general information and facts stored in the knowledge base[7].

One of the underlying aims of developing this functional online help system is to utilize the functional programming systems formality. The script control system matches the information list with the association list. If the attempt is successful the control of the dialogue is passed on to the script. However, if no existing script is appropriate for that query the system prompts the user for the next query. The structure content and use of a script are best explained through the use of an example. If a user were to ask the question: ***How can I see a calendar?***

The use of scripts provides a means by which users can do more than just access a help system that gives them textual information in a standard form leaving the user to decipher its often-cryptic content. Its intention is to animate the manual and for the system to act as an advisor rather than a reference. For example, when a user inputs the query: How can I see a calendar? Then, the best score may be 200 and the answer will instruct the user to use the certain command according to user's needs[8].

From that simple example, we can see the potential of the scripting system to provide a very useful extension to the knowledge-based component, filling the gap between the standard help facility and the human 'help desk' advisor.

4 Formulas for Computation

For the calculation in the natural language on line, it is necessary to introduce several formulas as follows:

Definition 1. A *probabilistic finite automaton* (PFA) A is 5-tuple A =(Q,$\sum$,δ ,q0,F) where Q is a finite set of states. $\sum$ is the finite input alphabet,q0 is the initial state viewed as a probability distribution over Q(e.g. q0= $\sum_{qi \in Q} aiqi$ with $\sum ai = 1$ and ai$\in$(0,1)), F$\subseteq Q$ is the accepting states, and δ is the

probabilistic transition function, that is , δ :Q×∑ ×Q→[0,1] satisfying $\sum_{q\in Q}\delta(p,\sigma,q)=1$ (1) for any p∈Q and σ ∈∑ .

The language recognized by above PFA A is defined to be a function LA: ∑ *→[0,1] as follows: for any x=$\sigma 1\sigma 2..\sigma k$ ∈∑ .*, where ∑ * stands for the set of all strings over ∑ , containing empty string ε ,LA(x)= ∑ {δ (q0,σ 1,q1)δ (q1,σ 2,q2)… δ (qk-1,σ k,q): q∈F, qi∈Q , I=1,2,…,k-1} (2) where δ (q0,σ 1,q1)= ∑ aiδ (qi,σ 1,q1) if q0=∑ aiqi.

A word over alphabet ∑ ={$\sigma 1\sigma 2..\sigma k$ } can be defined as a probability distribution W over ∑ , and it is denoted by W= a1\$\sigma$1 +a2\$\sigma$ 2+…+ak\$\sigma$ k, where W(σ i)=ai, I=1,2,…,k

Theorem 1. For any PFA A=(Q,∑ ,δ ,q0,F), and any string of words W=W1W2…Wk, we have

LA(W)= $\sum_{\sigma 1\sigma 2..\sigma k\in\sum} LA(\sigma 1\sigma 2...\sigma k)(\prod_{i=1}^{k}Wi)(\sigma 1\sigma 2...\sigma k)$, where ,also in what follows,$(\prod_{i=1}^{k}Wi)(\sigma 1\sigma 2...\sigma k)\overset{def}{=}$ W1(σ 1) W2(σ 2) …Wk(σ k)

We now describe an example to illustrate the application of Theorem 1.

Example 1. Let A=(Q,∑ ,δ ,q0,F) be a PFA, where Q={q1,q2,q3}, ∑ ={a,b}, q0=$\frac{1}{2}$q1+$\frac{1}{2}$q3,F={q3} and δ is defined as follows:

δ (q1,a,q1)=1/2, δ (q1,a,q3)=1/2
δ (q2,a,q2)=1/3, δ (q3,a,q1)= 1/3
δ (q3,a,q3)=2/3, δ (q1,b,q2)=2/3
δ (q2,b,q1)=1/4, δ (q2,b,q3)=1/4
δ (q3,b,q3)=1

otherwise , δ (p,x,q)=0. Then

LA(aa)= $\sum_{p\in Q}\delta(q0,a,p)\delta(p,a,q3)$ =43/72

LA(ab)= $\sum_{p\in Q}\delta(q0,a,p)\delta(p,b,q3)$ =7/12

LA(ba)= $\sum_{p\in Q}\delta(q0,b,p)\delta(p,a,q3)$ =1/3

LA(bb)= $\sum_{p\in Q}\delta(q0,b,p)\delta(p,b,q3)$ =7/12

Let the probability distribution W1,W2 over{a, b} be defined as;
W1=0.4/a+0.6/b, W2=0.1/a+0.3/b
Then, by Theorem 1 we obtain that

LA(W1W2)= $\sum_{\sigma 1,\sigma 2}W1(\sigma 1)W2(\sigma 2)LA(\sigma 1\sigma 2)$

=W1(a)W2(a)LA(aa)+W1(b)W2(a)LA(ba)+W1(a)W2(b)LA(ab)+W1(b)W2(b)LA(bb)
=197/900

In the same way, we can calculate, LA(W2W1)= $\sum_{\sigma 1,\sigma 2}W2(\sigma 1)W1(\sigma 2)LA(\sigma 1\sigma 2)$ =367/1800T

5 Concluding Remarks and Future Work

In this paper, we have considered some basic probabilistic models of computation by inputting strings of words, and some new conclusions have been discovered. Specifically, we have studied PFAs with input strings of words, and verified that PFAs computing strings of words can be implemented by means of calculating strings of symbols.

In reality, there are many significant computational models such as probabilistic neural networks, residuated lattice-valued automata, quantum automata.

Research in this area can be extended in both the development of the theory of NLP and in the development of functional programming as applied to the area of NLP in Web Pages. The two areas could be extended along the lines suggested by Hobbs and allow for the automatic generation of the functional equations of specified grammars, together with the automatic generation of scripts and templates from databanks as suggested in the emerging area of information extraction .

References

1. D. W. Qiu, H. Q. Wang, A probabilistic model of computing with words Journal of Computer and System Sciences 70 (2005) 176–200
2. J.R. Hobbs, Monotone decreasing quantifiers in a scope-free logical form, in: K.V. Deemter, S. Peters (Eds.), Semantic Ambiguity and Underspecification. CSLI Lecture Notes No.55, Stanford, California (1995)55-76.
3. R. Plant, S. Murrell, A natural language help system shell through functional , programming, Knowledge-Based Systems 18 (2005) 19–35
4. L .M. Adleman, Molecular computation of solutions to combinatorial problems, Science 266 (1994) 021–1023.
5. A. Ambainis, R. Freivalds, One-way quantum finite automata: strengths, weaknesses and generalizations, in: Proceedings of the 39th Annual Symposium on Foundations of Computer Science, Palo Alfo, CA, (1998) 332–341.
6. C. Runciman, N. Rojemo, Heap profiling for space efficiency, Second International School on Advanced Functional Programming, LNCS 1129, Springer, Olympia, WA, (1996) 159–183.
7. L.A. Zadeh, From computing with numbers to computing withwords-From manipulation of measurements to manipulation of perceptions, Ann. NY Acad. Sci. 929 (2001) 221–252.
8. D.W. Qiu, Automata theory based on complete residuated lattice-valued logic(I), Sci. China (F) 44 (6) (2001) 419–429.