The Design and Implementation of an Artificially Intelligent Personal Assistant Using Multiple Ontologies and Neural Ttoken Lists

Chun-Hsiang Hsieh
National Chung Cheng University
Taiwan, R.O.C.
hsiehch@cs.ccu.edu.tw

Daniel J. Buehrer
National Chung Cheng University
Taiwan, R.O.C.
dan@cs.ccu.edu.tw

Abstract:

We describe the design of an Artificially Intelligent Personal Assistant (AIPA) system whose theory is based on concepts from the artificial neural networks, triple-based knowledge representation that is used for both ontologies and their binary relationships, fast access via hash functions, and Class Algebra. The purpose of research and implementation is to study and test the effects of sharing multiple ontologies and Neural Token Lists (NTL), and to observe a Knowledge Sharing Connection (KSC) on the Internet, so that different knowledge of Closed Domains can be shared among humans and computer ontologies in order to significantly increase the knowledge content of the AIPA.

Keywords: Intelligent Assistant, QA system, Semantic Network, Ontology, Machine Learning.

1. INTRODUCTION

This research and implementation of an Artificially Intelligent Personal Assistant (AIPA) uses the following methods. First, we used the lexicon provided by the Chinese word segmentation system CKIP (Chinese Knowledge and Information Processing) [1] of Academia Sinica, Taiwan. Next, we did word segmentation by using the MM (Maximum Matching) method [2], and constructed an AIPA semantic network through the PDFT (phrase driven formal translation) method suggested herein, and built a Neural Token List. Then we processed the knowledge by using a HNN (Hopfield Neural Network) relational module [3]. Lastly, through our Knowledge Sharing Connection (KSC) technology, different computers using the AIPA system are able to share ontologies and NTL (Neural Token Lists) with other AIPAs. This allows different computers and users to learn and share Knowledge Tokens and their relationships as well as their data type information that is stored as triples in the ontologies.
A. QA System

A Question Answer (QA) system is a useful technology, especially suitable for searching for information or executing programs via natural language [4]. Generally, a QA system contains four processing stages- Question Analysis, Information Retrieval, Passage Selection and Answer Extraction. The users can input their questions colloquially and communicate with the system. The system can automatically use a variety of natural language processing techniques to respond to the users with relevant and correct answers quickly and efficiently.

B. Semantic Network

The building of a semantic network may associate objects or categories between related knowledge. For example, “A parrot is a bird,” entails an “is a” association between parrot and bird, and a “kind of” association between bird and animal. With associated reasoning through a semantic network, we can tell that a pigeon is “a kind of” animal. In addition, the description of other attributes of knowledge is also necessary, such as the appearance or shape of an object of a category (has shape), or that the property belongs to a certain object or a certain category (has property). For example, birds can fly (i.e. has a "has property" association to “can fly”).

II. ARCHITECTURE

There are three major parts in the AIPA program. One is the Knowledge Base, which contains words, the QSSDB/QASDB (Question Sample/Answer Sentence Databases) and SRADB (Sentence Relation/Action Database), the ontology, and the NTL (Neural Token List). The other part concerns association and learning. First, each sentence that is entered into AIPA is analyzed through Chinese NLP preprocessing, and the analyzed data is sent to the "Sentence Parsing Analysis Sentence Pattern Module", where the analysis of the sentence patterns is passed into the next PDFT (Phrase Driven Formal Translation) module, which analyzes the semantics. The Hopfield Neural Network (HNN) associates and recalls the ontology and NTL knowledge, and responds to users with inquired data. The third part uses the technology of a Knowledge Sharing Connection (KSC) to let different computers with AIPAs share their knowledge through an Internet connection, as shown in Figure 1.

Figure 1. The architecture of the AIPAs
A. Chinese NLP Preprocessing

1) Word Segmentation

This study adopted the MM (maximum matching) algorithm for word segmentation. It identified the permutations and combinations of each vocabulary phrase, and compared from left to right or right to left with the Lexicon, repeating these actions until it met the longest word permutation and combination. We consider this permutation and combination to be the best segmentation result.

Example:

Vocabulary: 北京大学 (Beijing University)
Step 1: 北/京大學; inquire for “北” in the Lexicon, answer: NA.
Step 2: 北京/大學; inquire for “北京” in the Lexicon, answer: Yes, reserved.
Step 3: 北京大/學; inquire for “北京大” in the Lexicon, answer: NA.
Step 4: 北京大學; inquire for “北京大學”, answer: Yes, and confirm this is the best result for word segmentation.

2) Sentence Parsing

AIPA does two parsing actions to a sentence: identifying the sentence pattern to determine whether it is an interrogative or declarative sentence, and analyzing the sentence pattern with a triple-mode parsing module.

a) Sentence Judgment

This research classifies Chinese sentences as either declarative or interrogative sentences. Since this is a mutually disjoint judgment, AIPA only judges against the interrogative sentences. Those sentences that are not interrogative are directly categorized as declarative. The way AIPA judges whether a sentence is interrogative is to analyze and conclude common sentence patterns by using the characteristics of Chinese interrogative sentences, and then build a QSSDB (Question Sentence Sample Database) where each sentence pattern is assigned into slots of triples. The slots are constructed as left-script, mid-script or right-script and each script may contain one or a pair of interrogatively described vocabulary. When a script contains NULL it means that such a description is dispensable. Finally, the system judges whether a sentence is interrogative by way of script matching.

QASSDB record Examples:
Slot: (NULL, 嗎/right?)
Slot: (NULL, ？)
Slot: (NULL, 有多少/how many, NULL)
Slot: (何時/When, NULL, NULL)
Slot: (何地/Where, NULL, NULL)
Slot: (哪/Which, NULL, NULL)
Slot: (怎麼/How, NULL, NULL)
Slot: (能夠/Can, 回應/respond, NULL)
Slot: (可以/Can, 回應/respond, NULL)
Slot: (請教/may I ask, NULL, NULL)
Slot: (請問/may I ask, NULL, NULL)
Slot: (什麼是/What is, NULL, NULL)

b) Sentence Analysis

The Sentence Relation-Action Database (SRADB) is used to divide a sentence or phrase into three parts: Active-Object, Relation-Action and Passive-Object. By putting all Relation-Actions in the database, these Relation-Actions include common Chinese relation-stop-words, relation-verb-words and relation-words. The
relation-stop-words and relation-verb-words are directly associated with the stop-words and verb-words recorded in the lexicon, while the relation-words are built by a rule base.

The way to match a Relation-Action is to scan the sentence from left to right; if there are vocabulary phrases consistent with Relation-Actions stored in the SRDB, the system separates out the Active-Object and Passive-Object.

SRADB record Examples:

Stop-word: 有 (have), 是 (be), 於 (at), 在 (at), 的 (’s), 之 (of), etc.

Verb-word: 闻 (smell), 等 (wait), 吃 (eat), 遮住 (cover), 設計 (design), 擦 (scrub), 擊出 (hit), 分析 (analyze), 增加 (increase), 預估 (estimate), 吸引 (attract), 走 (walk), 推出 (launch), etc.

Relation-word: 雖然 (although), 不管 (no matter), 喜歡 (like), 希望 (hope), 堅持 (insist), 出現 (appear), 隨著 (follow), 屬於 (belong to), 似乎 (seem), 不過 (but), 跟著 (with), 就是 (indeed), 除了 (beside), 落幕 (end), etc.

B. PDFT (Phrase Driven Formal Translation)

For a sentence, the system first analyzes the sentence pattern, and then determines whether it’s declarative or interrogative. Next, the sentence is converted into triples [5] and the corresponding vocabulary words are converted into non-terminals or terminals. These are stored into the slots of (object, event, object) triples or (object, attribute, object) triples. Each slot generated from the sentence has its own OID (Object ID) [6]. All of the slots are associated with each other and form a semantic network, including sub-class relationships and IS-A hierarchy associations.

Figure 2. The operations of the Triple mode
PDFT adopts the SRADB algorithm to disassemble the sentence into each slot (left-object, mid-status, right-object), until the left-object and right-object slots are all terminals, of which the left-object and right-object may store terminal or non-terminal objects. A non-terminal stands for a collection of objects that has not yet been determined, while a terminal stands for identified objects.

Mid-status may store descriptions of two kinds of objects: attributes and events. Attributes stand for the description (left-object) of other attributes (right-object) of knowledge, while events stand for the events (right-object) of the (left-object) knowledge. However, the terminal or attribute or event in each slot may independently or repeatedly use triples to define a semantic sub-network, as shown in Figure 2.

C. NTL (Neural Token List)

Lastly, AIPA looks up all terminals in the CKIP Lexicon. If the answer is correct, then AIPA identifies the terminal as "knowledge" (a Knowledge Token), and stores that Knowledge Token into a slot. All slots storing Knowledge Tokens form an NTL (Neural Token List), as shown in Figure 3.

The NTL describes AIPA's "longitudinal relations" and "lateral relations". For example, suppose that NTL stores three Knowledge Tokens, respectively, the U.S., a president and Obama. When we inquire the AIPA with "president", we can find the related knowledge "Obama". As for the "lateral relation", besides the interactive learning on the human-machine interface, Knowledge Tokens of an OID in the same layer of a slot in the NTL are compared, and two knowledge Tokens are considered laterally associated if they share an 80% similarity.

A HNN (Hopfield Neural Network) relational module is used to do the association and recalling from NTL and the ontology. In order to make sure that the recalling converges, when the Knowledge Token recalls the same OID, the recalling stops.
D. **KSC (Knowledge Sharing Connection)**

Each AIPA has its ontology that communicates with NTL to do association and recalling, and each ontology and NTL has a unique identification code called an OID. Through the OID, AIPA is able to process sub-class relationships and IS-A hierarchy associations. Using KSC technology, two or more different computers with AIPAs are able to communicate with each other, in order to communicate and learn.

The operating procedure of KSC is that AIPA cannot find the relevant terminal node and Knowledge Token when using its ontology and NTL; therefore, AIPA would move to next step and pass the request token to other AIPAs through an Internet connection. When they receive request tokens, they do association and recall via their ontology and NTL. If possessing relevant information, they would feed a response packet back, which would includes the terminal node, Knowledge Token, and AIPA ID, to the AIPA. If they find nothing, they do not respond to the AIPA.

When the AIPA receives the response packet, it decodes the response packet and saves the terminal node and Knowledge Token to a temporary ontology and NTL. Also, it uses the AIPA ID to categorize different sources. Finally, AIPA re-processes and does association and recall for the temporary ontology and NTL to find valuable information, as shown in Figure 4.

![Figure 4. The architecture of KSC](image)

### III. **Examples of Actual Operations**

- **AIPA(a)** receives a question from the users:
  
  User: 請問美國總統是誰？(Who is the U.S. President)

- If the AIPA(a) cannot find an answer in its system, it passes a request token to other AIPAs.

  **PDFT Learning:**
  
  Slot: 1.1 [oid] (美國 [terminal], 是 [attribute], 國家 [terminal])

  **NTL Learning:**
  
  Slot: 1.1 [oid] (美國 [terminal] in the Lexicon), is [Knowledge Token]
Slot: 1.2 [oid] (國家[terminal] in the Lexicon), is [Knowledge Token]
AIPA(a) : Request Token (『美國』(U.S.) / 『總統』(President))

- One AIPA(b) on the Internet automatically finds a similar terminal node and Knowledge Token.
  PDFT Learning:
  Slot: 1.1[oid]（美國[terminal], 是[attribute], 國家[terminal]）
  Slot: 1.2[oid]（歐巴馬[terminal], 是[attribute], 總統[terminal]）
  NTL Learning:
  Slot: 1.1 [oid] (美國[terminal] in the Lexicon), is [Knowledge Token]
  Slot: 1.2 [oid] (國家[terminal] in the Lexicon), is [Knowledge Token]
  Slot: 1.3 [oid] (總統[terminal] in the Lexicon), is [Knowledge Token]
  Slot: 1.4 [oid] (歐巴馬[terminal] in the Lexicon), is [Knowledge Token]

- Packs a terminal node and Knowledge Token response packet to respond to the AIPA(a).
  AIPA: Request Token(AIPA ID[ PDFT[Slot 1.1... Slot 1.2...]
  / NTL[Slot 1.1... Slot1.2... Slot 1.3... Slot 1.4...]])

- After receiving the response packet, the AIPA(a) decodes it. It saves the terminal nodes and Knowledge Tokens in its temporary ontology and NTL and uses theAIPA ID to organize where it is from.
  AIPA ID(b)

  Temporary PDFT:
  Slot: 1.1[oid]（美國[terminal], 是[attribute], 國家[terminal]）
  Slot: 1.2[oid]（歐巴馬[terminal], 是[attribute], 總統[terminal]）

  Temporary NTL:
  Slot: 1.1 [oid] (美國[terminal] in the Lexicon), is [Knowledge Token]
  Slot: 1.2 [oid] (國家[terminal] in the Lexicon), is [Knowledge Token]
  Slot: 1.3 [oid] (總統[terminal] in the Lexicon), is [Knowledge Token]
  Slot: 1.4 [oid] (歐巴馬[terminal] in the Lexicon), is [Knowledge Token]

- AIPA(a) re-processes and does association and recall in its existing and temporary ontology NTL in order to find relevant information.

  After NLP Preprocessing,
  PDFT Learning:
  Slot: 1.2[oid]（歐巴馬[terminal], 是[attribute], 總統[terminal]）
  NTL Learning:
  Slot: 1.4 [oid] (歐巴馬[terminal] in the Lexicon), is [Knowledge Token]
  Respond to the Question, as shown is Figure 5.
  總統(computer)／是(is)／歐巴馬(Obama) 總統是歐巴馬
IV. TEST RESULTS AND DISCUSSION

We used the same questions for AIPA to conduct three trials: 
A. Operating a computer with AIPA to answer. 
B. Operating two computers with AIPAs to answer. 
C. Operating three computers with AIPAs to answer.

A. Single Computer AIPA Test

First, we input 20 sets of information about travel into an AIPA system and tested 10 questions. The result was that two questions could not be identified, five questions were answered correctly and three questions were answered incorrectly. Next, we input another 10 travel-information sets to the AIPA and tested it with the same questions again. The result was that two questions could not be identified, six questions were answered correctly and two questions were answered incorrectly.

B. Double Computer AIPA Test

We marked a computer with AIPA system as AIPA(a) and another as AIPA(b), then used these computers to conduct the trial. We input different information for the AIPA(a) and the AIPA(b). For the AIPA(a), the same 20 travel information sets were input as in trial one. For the AIPA(b), 20 automobile information sets were input but tested exactly the same 10 travel questions as in trial one. The result was that two questions could not be identified, and eight questions were answered incorrectly. Next, we ran the KSC function of the AIPA network to share their ontology and NTL. We used the same questions to test the AIPA(b) and the result was that two questions could not be identified, five questions were answered correctly and three questions were answered incorrectly. Finally, we input 10 relevant information sets to AIPA(a) and tested AIPA(b) with the same 10 questions. The result was that two questions could not be identified, six questions were answered correctly and two questions were answered incorrectly.
C. Triple Computer AIPA Test

This time, we used three computers to conduct the trial. One computer with AIPA was named AIPA(a), another was named AIPA(b), and the other was named AIPA(c). We input different information for these AIPAs. For AIPA(a) we input the same 20 travel information sets as in trial one. For AIPA(b), we input the same 20 automobile information sets as in trial two. And for AIPA(c) food information sets were input. AIPA(b) was tested on the same 10 questions as in trial one and the result was: two questions were not identified and eight questions were answered incorrectly. Next, we started the KSC function of the AIPA network to share their ontologies and NTLs and tested AIPA(b) with the same questions. The result was that two questions could not be identified, five questions were answered correctly and three questions were answered incorrectly. Finally, we input 10 relevant information sets into AIPA(c) and then tested AIPA(b) with the same 10 questions. The result was that two questions could not be identified, six questions were answered correctly and two questions were answered incorrectly.

From the test results above we understand when an AIPA cannot find a similar terminal node and Knowledge Token, the system will pass requests to other AIPAs. Although a system has no relevant information in its ontology and NTL, it is able to bridge to other AIPA systems to obtain the knowledge it needs. Therefore, no matter if two or three computers, AIPAs are able to connect and share their terminal nodes and Knowledge Token through KSC technology. Closed Domains of different AIPAs can be exchanged by passing their terminal nodes and Knowledge Tokens to each other in order to greatly increase the content of the AIPAs knowledge.

REFERENCES

[1] THE CKIP CHINESE TREEBANK: GUIDELINES FOR ANNOTATION


