Online Multimodal interaction for Speech interpretation

Vaishali Ingle
M.E.(Computer Science)
Department of Computer Science Dr.Babasaheb Ambedkar Marathwada University, Aurangabad
Maharashtra(India)-(0240)2403316

Aditi Deshpande
M.Sc.(Computer Science)
Department of Computer Science Dr.Babasaheb Ambedkar Marathwada University, Aurangabad
Maharashtra(India)-(0240)2403316

ABSTRACT

In this paper, we describe an implementation of multimodal interaction for speech interpretation to enable access to the Web. As per W3C recommendation on 10th February 2009 the latest version of, EMMA is used for translation of speech signals into a format interpreted by the application language, greatly simplifying the process of adding multiple modes to an application. EMMA is used for annotating the interpretation of user input. The lattice is designed by considering the model, architecture, input modalities. The interpretation of the user's input is expected to be generated by signal interpretation process by speech.

Keywords

EMMA, SALT, X+V, speech, modality, annotation, Token, XPath, interpretation, lattice

1. INTRODUCTION

Extensible Multimodal Annotation markup language (EMMA) was developed in order to provide semantic interpretations for speech, natural language text, keyboard/, and ink input (a type of stylus input that includes handwriting recognition).

The widespread adoption of XML and derivative markup languages has, for all intents and purposes, enabled the advent of multimodal development. The three building-block languages for multimodal development are: SALT (Speech Application Language Tags), X+V (XHTML + Voice), and EMMA (Extensible MultiModal Annotation). All three have been submitted to the W3C for consideration as standards for telephony and/or multimodal applications. Currently, all three are under consideration for the next version of VoiceXML.

SALT: This language is an extension of HTML and other markup languages[2] (cHTML, XHTML, WML). It's used to add speech interfaces to Web pages and it's designed for use with both voice-only browsers and multimodal browsers—meaning, cellular phones, tablet PCs, and wireless PDAs.

Microsoft developed SALT specifically to enable speech across a wide range of devices and to allow telephony and multimodal dialogs. Because SALT uses the data models and execution environments of its host environments (HTML forms and scripting), it is more familiar to Web developers. Its event-driven interaction model is useful for multimodal applications.

However, SALT is merely a set of tags for specifying voice interaction that can be embedded into other "containing" environments. Because of this dependency on an external environment, developers using SALT may need to generate differing versions of an application for each device—for instance, an application for use on cell phones will require separate versions for Nokia and Motorola phones.

X + V: This IBM-sponsored language combines XHTML with VoiceXML 2.0, the XML Events module, and a third module containing a small number of attribute extensions to both XHTML and VoiceXML. This allows VoiceXML (audio) dialogs and XHTML (text) input to share multimodal input data.

The fact that X+V is built using previously standardized languages makes it easy to modularize—that is, to break apart its code into modes, where one mode is for speech recognition, one is for motion recognition, etc..

But using the XML Events standard is what really differentiates X+V from SALT. Whereas events drive the creation of X+V, thus defining the environment, SALT merely attaches its tags to events within a pre-existing environment. Because X+V is self-sufficient in this manner, applications written with it are generally more portable.

EMMA is a complimentary language to SALT and X+V, functioning as a sort of middleman between a multimodal application's components—that is, between a user's input and the X+V- or SALT-based interpreter. This frees developers from having to worry about writing code to interpret user input. EMMA simply translates input into a format interpreted by the application language, greatly simplifying the process of adding multiple modes to an application.

2. REQUIREMENTS FOR EMMA

EMMA is the markup language used to represent human input to a multimodal application. As such, it may be seen in terms of the W3C Multimodal Interaction Framework as the exchange mechanism between user input devices and the interaction management capabilities of an application.

2.1 General Principles

An EMMA document [1] can be considered to hold three types of data:

- Instance-data
  The slots and values corresponding to input information which is meaningful to the consumer of an EMMA document. Instances are application-specific and built by input processors at runtime. Given that utterances may be ambiguous with respect to input values, an EMMA document may hold more than one instance.

- Data-model
  The constraints on structure and content of an instance. The data model is typically pre-established by an application, and may be implicit, that is, unspecified.

- Metadata
  Annotations associated with the data contained in the instance. Annotation values are added by input processors at runtime.

Given the assumptions above about the nature of data represented in an EMMA document, the following general principles apply to the design of EMMA:

- The main prescriptive content of the EMMA specification will consist of metadata: EMMA will provide a means to express the metadata annotations which require standardization. (Notice, however, that such annotations may express the relationship among all the types of data within an EMMA document.)

- The instance and its data model is assumed to be specified in XML, but EMMA will remain agnostic to the XML format used to express these. (The instance XML is assumed to be sufficiently structured to enable the association of annotative data.)

- The following sections apply these principles in terms of the scope of EMMA, the requirements on the contents and syntax of data model and annotations, and EMMA integration with other work.

2.2 Scope and General Requirements

EMMA must be able to represent the following kinds of input**:** input in any human language , input from the modalities and input reflecting the results of the following processes, token interpretation from signal (e.g. speech + SRGS) ,semantic interpretation from token/signal (e.g. text + NL parsing/speech +SRGS+SI) input gained in any of the following ways: 1.single modality input 2. Sequential modality input, that is: single-modality inputs presented in sequence3. Simultaneous modality input 4. Composite modality input

2.3 Input modalities, devices and architectures

EMMA must be able to represent input from the following :

human language input modalities-text ,speech ,handwriting , other modalities identified by the MMI Requirements document as required ,combinations of the above modalities ,devices , telephones (i.e. no device processing, proxy agent) ,thin clients (i.e. limited device processing) ,rich clients (i.e. powerful device processing) ,everything in this range ,known and foreseeable network configurations ,architectures, protocols , extensibility to further devices and modalities [3]

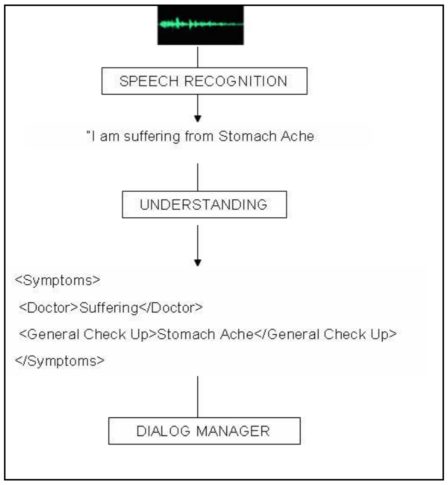
SPEECH RECOGNITION
"I am suffering from Stomach Ache
UNDERSTANDING
<Symptoms>
<Doctor>Suffering</Doctor>
<General Check Up>Stomach Ache</General Check Up>
</Symptoms>
DIALOG MANAGER

Figure 1 Speech Input Processing

2.4 Representation of output and other uses

EMMA is considered primarily as a representation of user input, and it is in this context that the rest of this document defines the requirements on EMMA. Given that the focus of EMMA is on meta information, sufficient need is not seen at this stage to define standard annotations for system output nor for general message content between system components. However, the following requirement is included to ensure that EMMA may still be used in these cases where necessary.

The following uses of EMMA must not be precluded:

- a representation from which system output markup may be generated;

- a language for general purpose communication among system components.

Ease of use and portability :EMMA content must be accessible via standard means (e.g. XPath).

- Queries on EMMA content must be easy to author.

- The EMMA specification must enable portability of EMMA documents across applications.

2.4.1 Data model requirements

- Data model content

The following requirements apply to the use of data models in EMMA documents

- use of a data model and constraints must be possible, for the purposes of validation and interoperability
- use of a data model will not be required
    - in other words, it must be possible to rely on an implicit data model.
- it must be possible in a single EMMA document to associate different data models with different instances

It is assumed that the combination and decomposition of data models will be supported by data model description formats (e.g. XML Schema), and that the comparison of data models is enabled by standard XML comparison mechanisms (e.g. use of XSLT, XPath). Therefore this functionality is not considered a requirement on EMMA data modeling.

2.4.2 *Data model description formats*

The following requirements apply to the description format of data models used in EMMA documents.

- Existing standard formats must be able to be used, for example:

    Arbitrary XML

    XML Schema

    XForms

- No single description format is required the use of a data model in EMMA is for the purpose of validating an EMMA instance against the constraints of a data model. Since Web applications today use different formats to specify data models, e.g. XML Schema, XForms, Relax-NG, etc., the principle that EMMA does not require a single format enables EMMA to be used in a variety of application contexts. The concern that this may lead to problems of interoperability has been discussed, and will be reviewed during production of the specification.
- data model declarations must be able to be specified inline or referenced

2.4.3 *Annotation requirements*

- **Annotation content**

EMMA must enable the specification of the following features. For each annotation feature[1], "local" annotation is assumed: that is, that the association of the annotation may be at any level within the instance structure, and not only at the highest level.

- **General meta data**
    - lack of input
    - uninterpretable input
    - identification of input source
    - time stamps
    - relative positioning of input events (NB: This requirement is covered explicitly by time stamps, but reflects use of EMMA in environments in which times tamping may not be possible.)
    - temporal grouping of input events
    - human language of input
    - identification of input modality
- **Annotational structure**
    - association to corresponding instance element annotated
    - reference to data model definition
    - Composite multimodal input: representation of input from multiple modalities.
- **Recognition (signal --> tokens processing)**
    - reference to signal
    - reference to processing used (e.g. SRGS grammar)
    - tokens of utterance
    - ambiguity
      This enables a tree-based representation of local ambiguity. That is, alternatives are expressible for given nodes in the structure.
    - confidence scores of recognition
- **Interpretation (tokens --> semantic processing)**
    - tokens of utterance
    - reference to processing used (e.g. SRGS)
    - ambiguity
    - confidence scores of interpretation

- **Recognition and Interpretation (signal --> semantic processing)**
  - union of Recognition/Interpretation features, (e.g. SRGS + SI)
- **Modality-dependent annotations**
  - EMMA must be extensible to annotations which are specific to particular modalities, e.g. those of: speech, handwriting
- **Annotation syntax**
  - The following requirements apply to the syntax that will be used in EMMA to express annotative data:
  - must enable association of annotations with instance data
  - must be compatible with RDF conceptual framework
  - must enable extensibility (optional/proprietary annotations)
  - (nice to have) may enable the specification of word graphs in addition to local ambiguity. NB - this is not currently seen as a necessary feature, and is unlikely to be sufficiently high priority to be addressed in the specification.

3. Implementation

In addition to providing the ability to represent N-best lists of interpretations using **emma:one-of**, EMMA also provides the capability to represent lattices of words or other symbols using the **emma:lattice** element. Lattices provide a compact representation of large lists of possible recognition results or interpretations for speech, pen, or multimodal inputs.

In addition to providing a representation for lattice output from speech recognition, another important use case for lattices is for representation of the results of gesture and handwriting recognition from a pen modality component. Lattices can also be used to compactly represent multiple possible meaning representations. Another use case for the lattice representation is for associating confidence scores and other annotations with individual words within a speech recognition result string.

Lattices are compactly described by a list of transitions between nodes. For each transition the start and end nodes MUST be defined, along with the label for the transition. Initial and final nodes MUST also be indicated. The following figure provides a graphical representation of a speech recognition lattice which compactly represents eight different sequences of words.

3.1 Lattice Element

In EMMA, a lattice[1] is represented using an element **emma:lattice**, which has attributes **initial** and **final** for indicating the initial and final nodes of the lattice. For the lattice below, this will be:

<emma:lattice initial="1" final="8"/>. The nodes are numbered with integers.

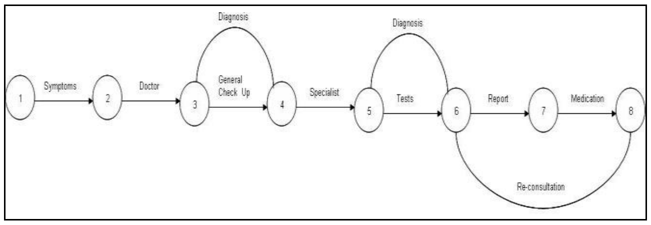

Figure 2 Lattice Diagram

which expands to:

a. symptoms Doctor General Check Up Specialist Tests Report Medications
b. symptoms Doctor Diagnosis Specialist Diagnosis Report Medications
c. symptoms Doctor Diagnosis Specialist Tests Report Medications
d. symptoms Doctor General Check Up Specialist Diagnosis Report Medications
e. symptoms Doctor General Check Up Specialist Tests Re-consultation
f. symptoms Doctor Diagnosis Specialist Tests Re-consultation
g. symptoms Doctor Re-consultation Specialist Tests Report Medications
h. symptoms Doctor Re-consultation Specialist Diagnosis Report Medications

There MUST only be one initial node in an EMMA lattice. Each transition in the lattice is represented as an element **emma:arc** with attributes **symptoms** and **medication** which indicate the nodes where the transition starts and ends. The arc's label is represented as the content of the **emma:arc** element and MUST be any well-formed character or XML content. In the example here the contents are words. Empty (epsilon) transitions in a lattice MUST be represented in the **emma:lattice** representation as **emma:arc** empty elements, e.g. **<emma:arc from="1" to="8"/>**.

3.2 Representation of EMMA Markup

```
<emma:emma version="1.0"
   xmlns:emma="http://www.w3.org/2003/04/emma"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.w3.org/2003/04/emma
    http://www.w3.org/TR/2009/REC-emma-
20090210/emma.xsd"
xmlns="http://www.example.com/example">
  <emma:interpretation id="interp1"
   emma:medium="acoustic" emma:mode="voice">
    <emma:lattice initial="1" final="8">
   <emma:arc from="1" to="2">Symtoms</emma:arc>

   <emma:arc from="2" to="3">Doctor</emma:arc>
   <emma:arc from="3" to="4">General Check Up</emma:arc>
   <emma:arc from="3" to="4">Diagnosis</emma:arc>
   <emma:arc from="4" to="5">Specialist</emma:arc>

   <emma:arc from="5" to="6">Tests</emma:arc>
   <emma:arc from="5" to="6">Diagnosis</emma:arc>
   <emma:arc from="6" to="7">Report</emma:arc>
   <emma:arc from="7" to="8">Medication</emma:arc>

   <emma:arc from="6" to="8">Re-consultation</emma:arc>
  </emma:lattice>
 </emma:interpretation>
</emma:emma>
```

4. Conclusion

The general purpose of EMMA is to represent information automatically extracted from a user's input by an interpretation component, [4] where input is to be taken in the general sense of a meaningful user input in any modality supported by the platform..

Components that generate EMMA markup:

1. Speech recognizers
2. Handwriting recognizers
3. Natural language understanding engines
4. Other input media interpreters (e.g. DTMF, pointing, keyboard)
5. Multimodal integration component

Components that use EMMA include:

1. Interaction manager
2. Multimodal integration component

Although not a primary goal of EMMA, a platform[5] may also choose to use this general format as the basis of a general semantic result that is carried along and filled out during each stage of processing. In addition, future systems may also potentially make use of this markup to convey abstract semantic content to be rendered into natural language by a natural language generation component.

5. REFERENCES

[1] EMMA:Extensible MultiModal Annotation Markup language W3C Recommendation 10th February 2009 http://www.w3.org/TR/2009/REC-emma-20090210

[2] Multimodality: Simple Technologies Drive a New Breed of Complex Application Input http://www.devx.com/wireless/Article/27878/1763

[3] An overview of EMMA—Extensible MultiModal Annotation-Michael Johnston,AT&T Labs Research http://www.research.att.com/~johnston/

[4] Multimodal Interaction Activity: http://www.w3.org/2002/mmi/

[5] EMMA: Extensible MultiModal Annotation 1.0: Implementation Report http://www.w3.org/2002/mmi/2008/emma-ir/#NotesOnTesting