Technology white paper

SILVIA's FUNCTIONAL ARCHITECTURE AND INTERACTIONS FLOW

Conversational Applications That Listen....and Talk

COGNITIVECODE®

White paper

SILVIA INPUT DECONSTRUCTION AND OUTPUT CONSTRUCTION

Table Of Contents

OVERVIEW

The SILVIA technology employs several unique algorithms to transform human language input into the n-dimensional conceptual space used by SILVIA's core, and to generate a set of possible "hits" in SILVIA's inputoutput blocks, otherwise known as behaviors. This collection of Input Deconstruction and evaluation techniques allows SILVIA to be flexible and organic in the way that she interprets user input and in the way that she derives sets of probabilistic IO blocks from that input.

Likewise the SILVIA technology employs multiple algorithms to transform SILVIA's conceptual output into human language. As with Input Deconstruction, this collection of patented techniques for Output Construction allows SILVIA to be flexible and organic, but as applied to the way that she assembles her responses to the user.

One key benefit to the collection of Input and Output components is the flexibility inherent in the system. Given even a small set of behaviors from which to work, SILVIA can immediately begin to exhibit emergent complexity in her ability to interpret meaning. Language rules are automatically inferred from the human language training data and from the probabilistic nature inherent in the algorithms.

By default, SILVIA never strictly enforces syntax rules while interpreting your input. Wish to talk like Yoda you do? Not a problem. SILVIA factors speaker idiosyncrasies into her equations, and will gladly interpret your meaning anyway. Of course, under mission-critical circumstances, one or more input filters may be flagged by a trainer as requiring strict syntax enforcement. "SILVIA, open the airlock please", would be an ideal candidate for such a flag.

OVERVIEW

The most critical technologies to the pre-processing effort lie in the production of the input/absorber coefficients. These algorithms first transform human language input into the conceptual structures that SILVIA's brain can use. Then, working in that concept-space, SILVIA can, in a context sensitive way, traverse the conceptual network of her knowledge base. She simultaneously traverses the localized conceptual network of the user's input to infer a resultant ordered n-length list or "stack" of most-likely matches from SILVIA's internal behavior and absorber data.

Once the Input Deconstruction phase has produced this ordered list, SILVIA can then derive her responses using the behavior data referenced by the list as a starting point. We will discuss the output in a different section of this white paper.

COGNITIVECODE®

PRE-PROCESSING

Before the above describe sub-system is invoked as a run-time process, there are certain pre-processing steps that must occur. The conceptual pre-processing is required for the algorithm to work in a balanced fashion, whereas the absorber (input pattern) pre-processing is mainly for purposes of optimization.

Concepts, Weights, and Roots

Using a series of algorithms, concept weights, and concept roots, SILVIA calculates a per-absorber value that represents an "ideal" matching input value. This pre-processing step is vital to the ability of SILVIA to rapidly find the best match of utterance to absorber. Conceptual pre-processing must be triggered anytime there has been a change in the base concept data. The absorber pre-processing must occur for newly created absorbers, when an absorber is modified, or if there has been a change in one of the statistical weights of an absorber's associated base conceptual data. Both conceptual and absorber pre-processing steps take place almost instantaneously, and can therefore be invoked as often as such changes occur without performance penalty. Again, note that by pre-processing the absorbers in this manner, significant performance improvements are achieved at runtime, as SILVIA can quickly discard absorbers or even entire behaviors where an input with a pre-calculated maximum falls far below the element's coefficient threshold.

If an absorber has "wildcards" for intercepting parts of user input, the wildcard symbols and captured portion of the input is not considered as part of the coefficient production process. For instance, if an absorber contains the phrase, "do you like *" and the user says "do you like talking to people", the coefficient will be produced by using only the two "do you like" portions of the two components.

PRE-PROCESSING

Once a particular input has been processed and tested with all valid absorbers, an n-length stack of the top "matches" will have been produced. This is an ordered list of objects or structures, depending on implementation. Each element of the list contains three components: a reference to the matching behavior, a reference to the matching absorber within the behavior, and the resultant master coefficient produced by the matching algorithm.

Given the above described list of top n matches, each behavior's validated exuders are tested for likely contextual matches. This check produces an additional aggregate weighting per exuder based on the number and types of context cues "consumed" by the exuder. This fractionally scaled aggregate is used as a modifier to the coefficient, allowing SILVIA to have an "attraction" to contextual cues and interesting behaviors, all other things being relatively equal. The fractional scale value is adjustable, and can be thought of as a slider that sets "importance of context" for SILVIA. If global contextual importance is low, SILVIA will tend to use the "winning" behavior and invoke the most contextually important exuder within that behavior. However, if importance is high, SILVIA may gravitate toward using a behavior that has a lower overall coefficient than the clear "winner", but has more relevant contextual cues embedded in one of its validated exuders.

If no valid behaviors are found to produce a response, depending on implementation, the AI can either not respond at all, or can respond in a one or more generic fashions to let the user know that it didn't understand the user, or was unable to formulate a response.

PRE-PROCESSING

Multiple Passes for Composite Concepts

The above described algorithm is sometimes invoked twice. The first pass is always performed as described above, with concepts serving as the base comparative units in the algorithm. The second pass may be performed with the following modification: instead of using root concepts as input, an algorithm attempts to re-map the concepts to an optimal collection of singular and compound concepts. If, after this re-mapping algorithm, the resultant input chain is different than the original singular concept chain, the full algorithm described above is run again with the new modified input. This difference is easily determined through a single array length comparison, because if the resultant "compound" chain is of a different length than the input, then two or more singular concepts have been re-mapped to a fewer number of compound concepts.

So, why do we re-run the entire input deconstruction algorithm again for these compound concepts? Because compound concepts can have relationships to other compound or singular concepts, and it is those relationships that are being tested. When a searchable n-dimensional tree is produced from both singular AND from those derived compound concepts, it allows for more variety in user input and accuracy in interpretation.

For instance, SILVIA may have learned a compound concept such as "go to the grocery store". She may also have learned that there is another compound concept, "travel to the market" that serves as a conceptual synonym to this first concept. They both convey a similar idea, yet when taken as their singular conceptual components, there are only two related (identical) concepts: to/to and the/the.

PRE-PROCESSING

Multiple Passes for Composite Concepts

So, if an absorber contains some variation on "go to the grocery store", if we never consider compound concepts and their mappings, a user input containing "travel to the local market" will produce a very low coefficient, even though in reality, the correlation in ***meaning*** is very high.

However, if we make sure to test the absorber having produced an ndimensional relationship tree containing as the base chain the user input's most likely used compound concepts, then a high correlation will be found in the above example, and that higher value will be added to the aggregate used to produce the coefficient.

OUTPUT

Using a variety of contextual cues, SILVIA uses a collection of "attraction" or "goal-seeking" algorithms to select likely knowledge block candidates from which to perform actions and construct her output, as well as to dynamically construct the output itself. Those contextual cues can include consumable events, topical context, or variable states.

Once the base candidate output filters, which we call "exuders", have been identified for constructing a response, depending on various contextual factors, SILVIA can invoke a variety of methods in order to turn this base information into a final human language output. Along the way, she is able to reference other knowledge blocks as "expert" information.

As with Input Deconstruction, language rules are automatically inferred from the human language training data and from the probabilistic nature inherent in the algorithms. However, the methods employed are, of necessity, fundamentally different. Instead of extracting meaning from human language input, SILVIA must, using one or more human language filters, dynamically construct a response that expresses the necessary output concepts in a meaningful and grammatically correct fashion.

To do this, SILVIA uses a hybrid of a heavily extended and modified Hidden Markov Model algorithm, and a conceptual substitution algorithm to produce potentially endless variety in output. In practice, with smaller numbers of behaviors, exuders, and conceptual associations, SILVIA will tend to respond using existing exuders almost "as is". However, as her knowledge base grows large enough, the additional resources begin to give SILVIA more "dynamic range". This is because SILVIA is able to infer larger numbers of probabilistic paths to express a given set of concepts while maintaining lexical correctness.

CONTEXT SENSITIVITY

At its simplest, SILVIA's context sensitivity during output is derived from a combination of exclusion, attraction, and modification based on the current context.

What do we mean by "the current context"? It is the current state of SILVIA's brain, including conversational history, events, variable values, and other data representing this state. For example, when training SILVIA, a common task is to attach explicit context cues to an exuder, so that depending on SILVIA's conversational history, different responses could be invoked contextually from the same input. Aside from creating behaviors, absorbers (our shorthand for user input filters), and exuders, a large part of an application trainer's job is leveraging context cues, the event system, variables, scripting, and the API to control SILVIA's behaviors in an application-specific way.

Exclusion

Before Output Construction can begin, the Input Deconstruction phase cherry-picks likely knowledge blocks. However, an exclusion filter is used to simultaneously REMOVE any output filters (exuders) from consideration if they cannot possibly be used in the current context. If no exuders within a particular knowledge block (behavior) are valid within the current context, then that entire behavior is discarded from consideration.

For each exuder, exclusion can occur if one or more required events does not exist on the queue, if one or more variable conditions have not been met, if one or more required contextual cues do not exist, or if a variable used by the raw exuder content does not exist. Also, an entire knowledge block (behavior) can be discarded from consideration if its named group is not active in memory, or if it has no input filters (absorbers). I will describe SILVIA's usage of "absorber- less" behaviors later in this document.

CONTEXT SENSITIVITY

Attraction

For SILVIA, attraction takes several forms, but in this instance, we are talking about SILVIA's attraction to context or to interesting behaviors contained in the remaining non-excluded exuders. Basically, SILVIA is more likely to use an exuder for output construction if it is consuming one or more events, if it has explicit contextual cues learned through training, if it invokes events or commands, or if the raw exuder data makes use of variables.

However, all other things being equal, SILVIA is also capable of using implicit context cues so that, even without explicit contextual training, she can be more "attracted" to output filters that either have some topical relevance, or that contain important conceptual as-yet unexpressed "thoughts".

Modification

Once one or more "winning" exuders have been selected through exclusion and attraction, we can then use the raw exuder concept data as a base to construct and modify a final response that the user sees, or for more rigid mission-critical responses, use the exuder(s) “as-is” with little to no modification.

CONCEPT SUBSTITUTION

Once SILVIA has constructed a reasonable response using the methods described above, she is almost ready to say what's on her mind. However, in a manner similar to Input Deconstruction, SILVIA then uses a two-pass algorithm to introduce even more potential variety in her output.

First, using similar methods as used by the Input Deconstruction algorithm, SILVIA attempts to re-map the output's base concepts to an optimal collection of singular and compound concepts.

Next, for each singular or composite concept in newly remapped output, SILVIA looks for lexical synonyms. A lexical synonym is one that can ALWAYS be used as a direct one-to-one replacement in the output's conceptual array. For instance, "travel to the grocery store" and "go to the market" can be considered lexical synonyms because you can swap one for the other in almost any conceivable circumstance.

If there are lexical synonyms to a particular concept, SILVIA randomly chooses from the collection of two or more lexical synonyms, including the source concept. If the random selection is a different value than the original source concept, SILVIA replaces it in the chain. The probability of a particular lexical synonym being used is based on actual usage statistics in SILVIA's overall content. This way, in a manner that is similar to her special HMM implementation, she has a probabilistic tendency to use language and concepts that are more commonly represented in her exuders, whereas less used synonymous concepts and phrases will appear less often in her output.

Now, any composite concepts remaining in the output are split back out into their singular conceptual components. SILVIA then re-runs the above described random lexical synonym substation algorithm on those base components. This means that not only do composite concepts have a chance to get replaced, but that each singular concept within any composite concept has the same opportunity.

FINAL OUTPUT

After SILVIA has done everything described in this document to produce the output, she has in her possession a numeric representation of a grammatically correct final output. All that remains is for her to apply a relatively simple concept-to-text conversion algorithm to create human readable text. Concurrently, she also replaces any textually embedded variables with the values contained in those variables. If the variable is a "complex" variable (see "Events and Commands"), in that it contains more than one value simultaneously, SILVIA replaces the reference to the variable with only one of the possible hidden values. The final resultant text is either directly output to the user, spoken by SILVIA, or both.