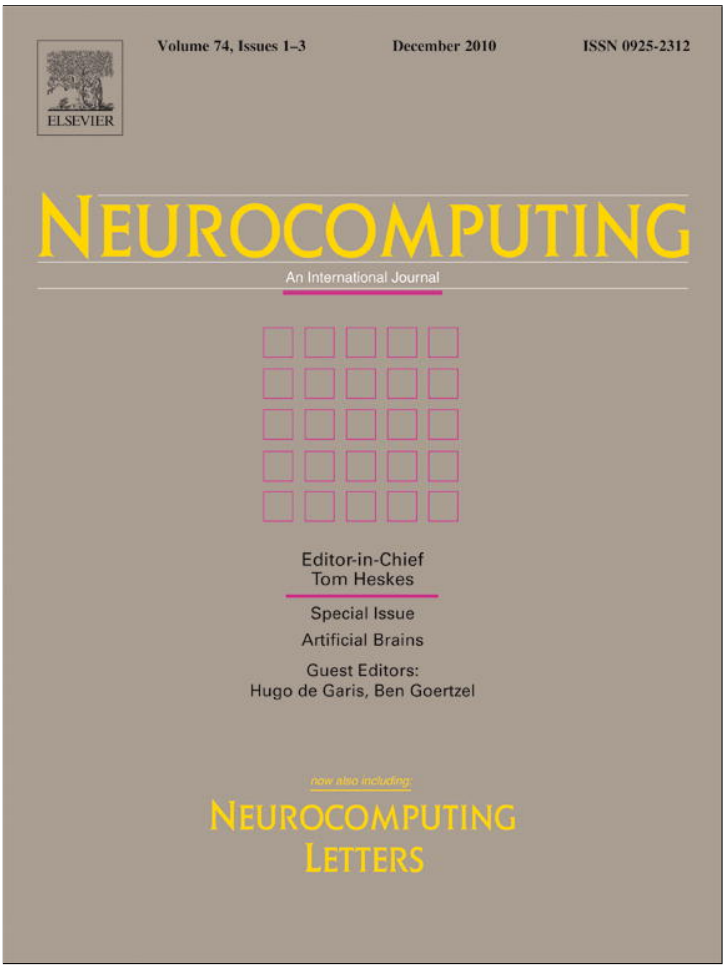

Neurocomputing 74 (2010) 95–103

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Sentence generation for artificial brains: A glocal similarity-matching approach

Ruiting Lian [a,*], Ben Goertzel [a,b], Rui Liu [b], Michael Ross [a], Murilo Queiroz [b], Linas Vepstas [b]

[a] Fujian Key Lab of the Brain-like Intelligent Systems, Xiamen University, Xiamen, China
[b] Novamente LLC, 1405 Bernerd Place, Rockville MD 20851

ARTICLE INFO

Available online 14 July 2010

Keywords:
Sentence generation
Language generation
Artificial brains

ABSTRACT

A novel approach to sentence generation – SegSim, Sentence Generation by Similarity Matching – is outlined, and is argued to possess a number of desirable properties making it plausible as a model of sentence generation in the human brain, and useful as a guide for creating sentence generation components within artificial brains. The crux of the approach is to do as much as possible via similarity matching against a large knowledge base of previously comprehended sentences, rather than via complex algorithmic operations. To get the most out of this sort of matching, a certain amount of relatively simple rule-based processing needs to be done in pre- and post-processing steps. However, complex algorithmic operations are required only for the generation of sentences representing complex or unfamiliar thoughts. This, it is suggested, is the sort of sentence generation approach that makes sense in a system that – like a real or artificial brain – combines the capability for effective local application of logical rules with the capability for massively parallel, scalable, inexpensive similarity matching.

1. Introduction

The process of language generation involves a series of stages [3,4], which may be defined in various ways, such as:

- *Content determination*: figuring out what needs to be said in a given context;
- *Discourse planning*: overall organization of the information to be communicated;
- *Lexicalization*: assigning words to concepts;
- *Reference generation*: linking words in the generated sentences using pronouns and other kinds of reference;
- *Syntactic and morphological realization*: the generation of sentences via a process inverse to parsing, representing the information gathered in the above phases; and
- *Phonological or orthographic realization*: turning the above into spoken or written words, complete with timing (in the spoken case), punctuation (in the written case), etc.

All of these stages are important, and there is a nontrivial amount of feedback among them. However, there is also a significant amount of autonomy, such that it often makes sense to analyze each one separately and then tease out its interactions with the other stages. Here we focus on the single stage of "syntactic and morphological realization," which we refer to for simplicity as "sentence generation" (taking a slight terminological liberty, as "sentence fragment generation" is also included here).

Sentence generation may be achieved in many ways; the variety of algorithms in the literature [5–8] barely scratch the surface of the scope of possibilities. However, when one thinks about how sentence generation might most feasibly be achieved in a brain or brain-like system, the range of possibilities narrows somewhat. Brains are not particularly good at carrying out complex serial algorithms requiring explicit exploration of large search trees. On the other hand, they are very good at carrying out approximate similarity matching of target items against large sets of remembered items. Thus, the most neurally feasible sentence generation approach would be the one that eschews complex algorithmic search processes whenever possible, in the favor of massively parallel similarity matching.

But how might this work? Our approach, which we label SegSim, is relatively simple and is given as follows:

1. The NL generation system stores a large set of pairs of the form (semantic structure, syntactic/morphological realization);
2. When it is given a new semantic structure to express, it first breaks this semantic structure into natural parts, using a set of simple syntactic-semantic rules;

* Corresponding author.
E-mail address: lianlian1022@gmail.com (R. Lian).

0925-2312/$ - see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.neucom.2009.11.053

3. For each of these parts, it then matches the parts against its memory to find relevant pairs (which may be full or partial matches), and uses these pairs to generate a set of syntactic realizations (which may be sentences or sentence fragments);
4. If the matching has failed, then (a) it returns to Step 2 and carries out the breakdown into parts again. But if this has happened too many times, then (b) it recourses to a different algorithm (most likely a search or optimization-based approach, which is more computationally costly) to determine the syntactic realization of the part in question;
5. If the above step generated multiple fragments, they are pieced together, and a certain rating function is used to judge if this has been done adequately (using criteria of grammaticality and expected comprehensibility, among others). If this fails, then Step 3 is tried again on one or more of the parts; or Step 2 is tried again. (Note that one option for piecing the fragments together is to string together a number of different sentences; but this may not be judged optimal by the rating function.); and
6. Finally, a "cleanup" phase is conducted, in which correct morphological forms are inserted, and articles and certain other "function words" are inserted.

Here we describe the SegSim approach from several perspectives, first presenting a few thoughts on how some process resembling SegSim might be realized in the human brain, and then describing a computational implementation of the SegSim approach that we have created, making use of several open-source computational linguistics tools. We also briefly describe how one would go about creating a connectionist implementation of SegSim, which might serve as a passable model of relevant parts of the brain as well as a useful computational system. Finally we give some concrete examples of sentences generated by our (non-connectionist) implementation of SegSim, which we call SegSim Prime.

While it is interesting to explore SegSim on its own, the real value of the approach lies in its easy integration with other cognitive and neural processes. SegSim was created partly in order to explain sentence generation in the human brain, and more critically in order to provide a practical approach to sentence generation in artificial brains. NLGen was created as part of the OpenCog AI architecture [2], which is heavily cognitive science inspired but does not seek to emulate the lower-level structures and dynamics of the brain; but it is now being developed in coordination with Xiamen University's Conscious Robot Project, which incorporates an effort to create a somewhat more realistic artificial brain. At the end of the paper we will briefly describe the integration of NLGen with these systems.

We emphasize throughout one of the key conceptual principles embodied by the SegSim approach, which is glocality [1]; the synthesis of global and local memory and processing. Glocality is embodied in the synergy between Steps {2 and 4} and 3 in the above algorithmic outline.

In Step 3, a widely distributed global memory and processing approach is utilized, in which a semantic structure is matched against a large body of previously encountered (semantic structure, syntactic/morphological realization) pairs. In Steps 2 and 4, a localized memory and processing approach is utilized, involving algorithmic application of explicitly stored grammatical rules. We suggest that this harmonious interoperation of these different types of memory is a critical aspect of sentence generation. Each localized memory item (each semantic substructure created in Step 2) must serve as a key for unlocking a globally distributed collection of memories (previously interpreted sentences).

2. Sentence generation in the brain

Contemporary neuroscience is not yet at the level where it can tell us exactly how sentence generation occurs in the brain. However, by piecing together knowledge about sentence generation in particular with knowledge about brain function in general, it is possible to formulate a reasonable set of conjectures regarding the broad manner in which neural sentence generation most likely occurs. In this section we will review some of the relevant facts from neuroscience, and use them to drive some conjectures (Fig. 1).

2.1. Known and hypothesized neuroscience of sentence generation

Sentence generation is a complex process involving multiple complex dynamics in multiple regions of the brain. However, there is a significant evidence [9] that a dominant role is played by area BA (Brodmann Area) 45 within the inferior frontal gyrus, a brain region sometimes grouped with BA 44 into "Broca's Area." This brain region is differentially activated when individuals need to express content syntactically rather than merely using disorganized or ordered sets of words. BA 45 is also involved with a variety of other reasoning functions including metaphorical and some reasoning processes (and also with some apparently unrelated processes including identification of familiar smells). However, other studies [10] have found that BA 44/47 are apparently involved with syntax processing related to inflections, whereas BA 45 is not. So it is not yet clear exactly how the various functionalities related to sentence generation are distributed among these nearby brain areas.

There is also evidence [11] that the same brain regions at or near areas BA 6, 9, 44, 45 and 46 are responsible for both working-memory and long-term-memory access. This is consistent with the notion that sentence generation involves dynamic working-memory/long-term-memory integration.

Fiebach and coworkers [12,13] have proposed a unifying neurological model explaining the role of Broca's area and the ventral premotor cortex in the general predictive processing of hierarchical sequential information. In this model, the brain carries out syntax processing via utilizing more generic mechanisms whose overall purpose is the prediction of sequential data that has hierarchical structure. Sentence generation may be viewed as an instance of a prediction problem, in that, given a set of semantic content; one may pose the prediction problem of predicting which word should come next in the generated sentence.

They propose the existence of a general neural mechanism for binding the temporally segregated sub-entities of a series of events into a single unified entity. The paradigm case of this in language processing is the grouping of a series of words into a phrase or clause. They then propose some quite specific characteristics of sentence processing as carried out by these brain regions:

These results suggest PMv/Broca involvement whenever the structure of a sentence deviates from a preferred structural template (defined according to grammatical rules and simplicity considerations). Within this region, distinct subregions appear to respond more specifically to different kinds of deviations—indicating they may be involved in different aspects of on-line syntactic processing.

> We suggest that (a) PMv/Broca region is involved in analyzing sentences according to grammatical rules, and that (b) the complexity of rule-based processes carried out in the PMv/Broca region varies along a posterior-to-anterior gradient. More posterior subregions perform an initial check concerning

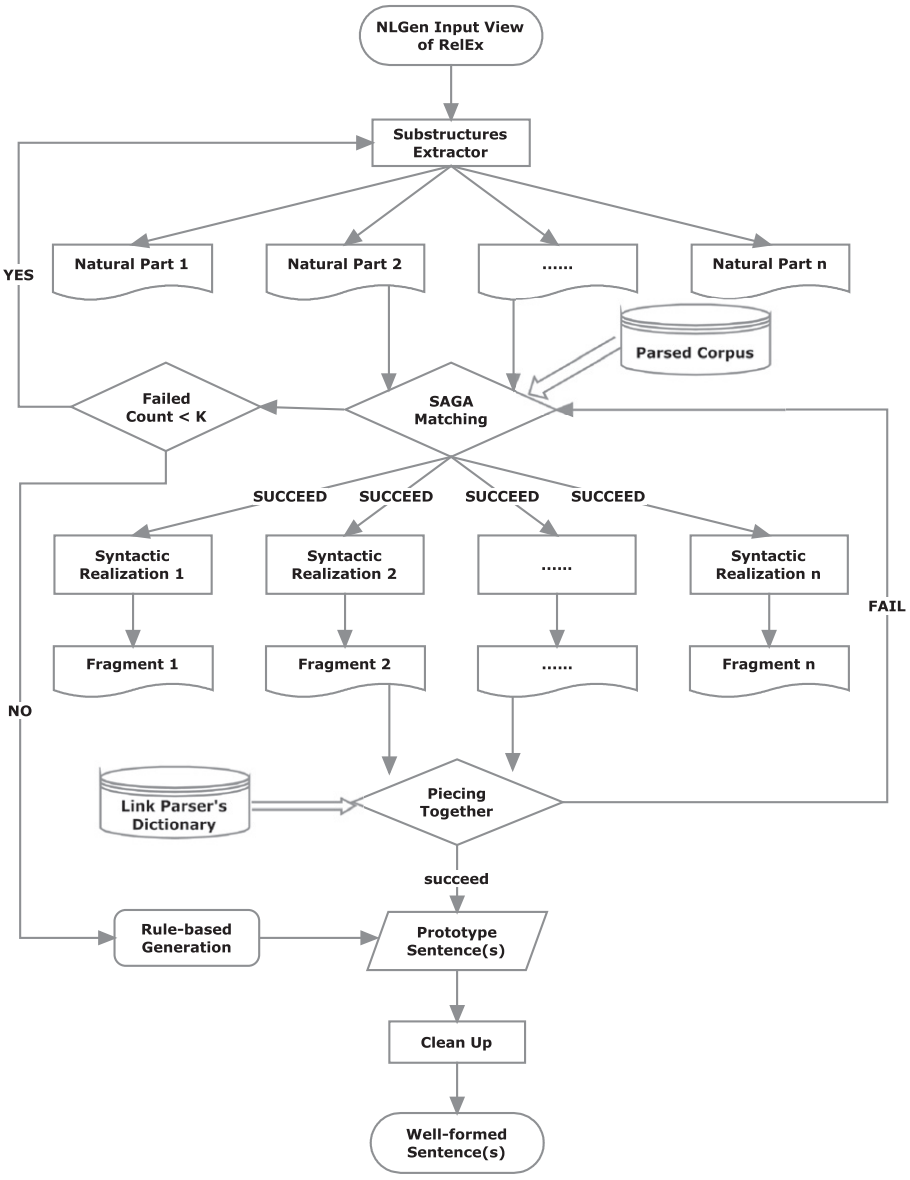

Fig. 1. Framework of NLGen.

the compatibility of the input with grammatical rules, which might be best described as mapping the input onto structural templates of simple and frequent sentence structures. More anterior regions (BA 44 and 44/45) utilize these rules to adjust syntactic predictions when processing complex sentences and to relate grammatically more complex sentence structures back to the simplicity-based structural templates.

While phrased largely in the language of sentence comprehension, these proposals are equally relevant to generation, if one assumes a perspective (like the one we advocate below) in which generation involves beginning with semantic content and then producing a sentence conditioned on that semantic content, via iteratively predicting the most likely word in the sentence based on the semantic content and the previous words. In the context of sentence generation, the qualitative hypothesis we may draw from these ideas is that:

- Generation of sentences according to simple templates may be its own coherent process and
- Generation of sentences that deviate from simple templates may involve a separate, coupled process that relies on a variety of different subprocesses, handling sentences of differing degree and type of complexity.

While we consider this an important and stimulating hypothesis, our inclination is to push back on the breadth of use of "rules" and "templates" here. Rather than relying primarily on explicit sentence-structure rules templates, we suggest, the brain may combine the use of simple explicit templates, with a different sort of process that involves similarity-matching against a large pool of already-understood example sentences stored in long-term memory. This alternate interpretation appears equally harmonious with the empirical results that Fiebach et al. draw on.

We do not wish to overstate our point here: it seems likely to us that the brain's language processing dynamics does involve some use of abstract rules and representations, especially in the processing of complex and unfamiliar sentences. Also, we are not advocating a simplistic similarity matching on the level of

phonemes, but rather a subtler similarity matching on the level of semantic and syntactic relationships.

But, a key conceptual point we wish to emphasize is that the brain's architecture is innately better suited for massively parallel similarity matching than for complex recursive abstraction [25]. This makes it natural to hypothesize that the brain should rely on simple similarity matching approaches wherever possible (in language generation and elsewhere). With this in mind it is interesting to explore how far one can go in language generation via relatively simple similarity matching approaches; and the answer we have arrived at via our work with NLGen is: rather far!

So, in our hypothesis, introducing the language of glocal memory theory [1], the regions within PMv/Broca that Fieback et al. interpet to contain complex rules for handling complex sentence structures, may actually serve as "keys" that open a "lock" consisting of a set of long-term memories of sentence/interpretation pairs matching the template. That is, the role of PMv/Broca may be twofold:

1. To apply a small set of template rules for interpreting very simple, frequently encountered syntax/semantics correspondences;
2. To handle more complex constructs via using these templates to guide the matching of sentence-fragments or semantic information against a large corpus of examples stored in LTM; and
3. To handle even more complex or unfamiliar constructs via invoking some sort of more algorithmic, recursive sentence generation process.

2.2. Connectionist models of sentence generation

Largely separately from the above ideas and observations from neuroscience, a number of AI researchers have proposed connectionist models of sentence generation and comprehension.

For instance, Kalita and Shastri [14,15] created a relatively simplistic system focused on the problem of producing the words in a subject–verb–object sentence given the thematic role fillers and indications of the desired voice and tense. Their approach seems unlikely to be extendable to sentences with recursive structure. Gasser's Connectionist Lexical Memory [16] is somewhat more ambitious, using a system in which bindings to syntactic roles are encoded with synchronized firing, but also does not produce recursive structures or handle long-distance dependencies. Miikkulainen [17] presented neural networks capable of producing multi-clause sentences based on a slot filler representation of its clauses, but these networks also seem to lack general generative capability, though their generality could possibly be extended via various improvements.

Perhaps the most ambitious connectionist sentence generation system is Rodhe's [18] Connectionist Sentence Comprehension and Production (CSCP) model. In Rohde's model, semantic relationship sets are represented using sets of propositions, which are encoded using distributed, featural representations. One portion of the neural network, the semantic system, is trained via the requirement to answer fill-in-the-blank questions about the propositions, and thus learns to compress a sequence of these propositions into a single, static representation of the meaning they embody. Another portion of the network handles comprehension, via learning to receive a sequence of words, encoded in a distributed phonological representation, and to output a representation of the sentence meaning in a format interpretable by the semantic system. The system is trained to learn the grammar; and using cases in which the meaning is provided in advance, it learns to make accurate predictions of which word will occur next in a sentence, using a combination of syntactic and semantic information. Sentence generation is then done by explicitly reversing comprehension: In order to generate a sentence, the neurons representing semantic meaning are clamped to fixed values representing the content to be expressed, and the network is asked to successively predict the words in the sentence.

We find Rohde's work impressive and conceptually agreeable, and consider our own work to be in a similar spirit. However, we have come at the problem from a slightly different direction, being motivated in our own work largely by the desire to create a maximally functional sentence generation system for practical utilization in the context of integrated artificial brains and AI systems. Thus, the system we have crafted has adopted a somewhat different architecture, driven partly by theoretical considerations and partly by the practicalities of working with existing (non-connectionist, but highly functional) computational linguistics software. This work has then led us to some different suggestions regarding how one might create connectionist sentence generation systems, which we will discuss a little later on.

Comparing Rohde's ideas to the hypotheses of Fiebach et al. reviewed above, we note here the absence of explicit templates embodying syntactic structures. Rather, one simply trains networks with (sentence, interpretation) pairs, and then the network learns the appropriate mapping. Comprehension and generation are then carried out by clamping inputs corresponding to the sentence or the interpretation respectively (to simplify just a bit). This is a purely distributed model, highly elegant in nature. However, we suggest that Fiebach et al. may have something valuable in their suggestion of simple template rules coupled with alternate processing for handling more complex cases. In the glocal approach we outline below, simple grammatical principles are used to break sentences into chunks, which are then remembered in a manner that facilitates rapid matching against fragments of semantic relationship-sets in need of expression—an approach which combines a simple, localized template-based approach like that to which Fiebach et al. allude, with a distributed, similarity-matching based approach as Rohde uses.

2.3. Large-scale similarity matching as a key aspect of brainlike algorithms

Rodhe's excellent work showcases one of the great strengths of recurrent neural network architectures and of the human brain, noted above: the capability for rapid, approximate similarity matching across large knowledge bases. Rohde's network achieves this via particular learning algorithms, and Hopfield nets and their descendants [19] achieve the same thing via different algorithms. In both cases the core message is the same: these sorts of systems are very good at solving problems of the form "Given a query Q and a large number of stored memories find the stored memories that best match Q."

The brain very effectively deploys its massively parallel architecture to achieve this sort of matching efficiently. On the other hand, implementing neural nets on von Neumann machines, as in the case of Rohde's algorithm, obviously does not offer the same computational advantages. In fact, one may argue that when crafting implementations on standard contemporary computing hardware, it may be better to implement large-scale similarity matching via some other algorithm, not closely based on the brain but more closely adapted to the underlying hardware. Regardless of the particular implementation, though, the point remains that in many cases, for an algorithm/architecture to be "brainlike" on a conceptual level, the correct course is to rely less on complicated searching or analytical algorithms, and more on simplistic but large-scale similarity matching against stored information.

3. A computational algorithm for sentence generation via large-scale glocal similarity matching

The SegSim sentence generation process described in the Introduction above is quite broad and could be implemented in a variety of different ways. In this section we describe the one implementation with which we have experimented, NLGen, which is based on the RelEx language processing system [20] that incorporates the Carnegie–Mellon link parser [21].

The particular approach described in this section is not connectionist in nature; however, it is still strongly neurally inspired in its overall design and conception; and, after presenting the particulars of the algorithms involved, we will give some comments about how it might be realized in a connectionist manner if one wished.

A note on why we have not taken a connectionist implementation approach, in spite of the neural inspiration of our algorithm, may be worthwhile. The main reason is that our goal has been to draw on neural inspiration to create a highly effective practical NL generation system. And Rohde's work does involve some similarity matching, which is conceptually similar to our approach; but his design doesn't provide a scalable mechanism for similarity matching, nor give a means for similarity matching to cooperate with more advanced mechanisms like Chomsky's Merge [24]. In line with some comments made above, our contention is that, while a connectionist implementation might be of intellectual interest, due to efficiency considerations it would be unlikely to be the best approach in terms of practical computational linguistics performance on contemporary commodity hardware. Furthermore, our knowledge of low-level brain mechanisms is insufficient to allow us to create a truly biologically realistic implementation of sentence generation, if this were our desire.

Recall from the Introduction that SegSim has 5 major aspects:

1. Storage of (syntactic structure, semantic structure) pairs;
2. Decomposition of the target semantic structure into a set of substructures;
3. Matching of these semantic substructures against the datastore of pairs;
4. Assembly of fragmentary syntactic structures into series of sentences; and
5. Cleanup, involving insertion of morphology and various syntactic markers.

Some specific details will be discussed in Section 3.2 below, after some preliminary material.

3.1. The RelEx framework for natural language comprehension

RelEx is an English-language semantic relationship extractor, which consists of two components: the dependency extractor and the relationship extractor (see Fig. 2). It can identify subject, object, indirect object and many other dependency relationships between words in a sentence; it generates dependency trees, resembling those of dependency grammars.

The dependency extractor component carries out dependency grammar parsing via a customized version of the open-source Sleator and Temperley's link parser [21]. The link parser outputs several parses, and the dependencies of the best one are taken [20]. The "Link Parser Representation" is the output of the Link Parser [21], which is documented on the link parser's website.[1]

[1] http://www.link.cs.cmu.edu/link/

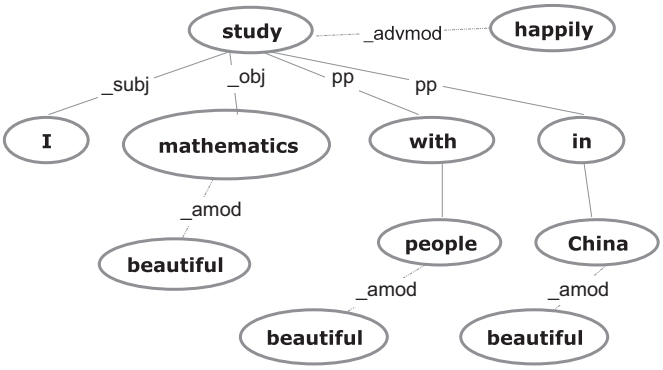

Fig. 2. Framework of RleEx.

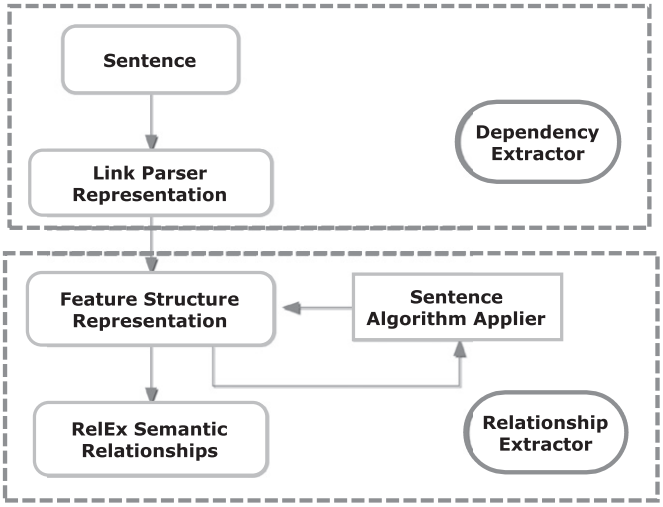

Fig. 3. Example of a substructure.

The relationship extractor component is composed of a number of template matching algorithms that act upon the link parser's output to produce a semantic interpretation of the parse[20]. It contains three steps:

1. Convert the Link Parser output to a feature structure representation.
2. Execute the Sentence Algorithm Applier which contains a series of Sentence Algorithms to modify the feature structure.
3. Extract the final output representation by traversing the feature structure.

A feature structure is a directed graph in which each node contains either a value, or an unordered list of features. A feature is just a labeled link to another node. Sentence Algorithm Applier loads a list of Sentence Algorithms from the algorithm definition file, and the Sentence Algorithms are executed in the order they are listed in the file. RelEx iterates through every single feature node in the feature structure, and attempts to apply the algorithm to each node. Then the modified feature structures are used to generate the final RelEx semantic relationships.

3.2. The NLGen framework

As described above, NLGen is an implementation of the SegSim concept that focuses on sentence generation from RelEx semantic relationships, as part of the OpenCog AI architecture [2].

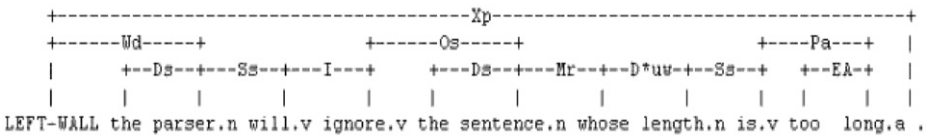

Fig. 4. Linkage of an example.

In NLGen, Step 1 is handled in a very simple way using a relational database. In a later version this may be modified to utilize the AtomTable semantic network data store provided by the OpenCog integrated AI framework [2].

The substructure we used in Step 2 is defined by the predicates of the sentence, i.e. we define one substructure for each predicate, which can be described as follows:

Predicate (Argument$_i$ (Modify$_j$))

where $1 =< i =< m$ and $0 =< j =< n$ and m, n are integers; "Predicate" stands for the predicate of the sentence, corresponding to the variable $0 of the RelEx relationship _subj($0, $1) or _obj($0, $1); Argument$_i$ is the ith semantic parameter related with the predicate; Modify$_j$ is the jth modifier of the Argumenti.

For instance, given the sentence "I happily study beautiful mathematics in beautiful China with beautiful people." The substructure can be defined as Fig. 3.

If there is more than one predicate, then multiple subnets are extracted analogously.

For each of these substructures, Step 3 is supposed to match the substructures of a sentence against its global memory (which contains a large body of previously encountered [semantic structure, syntactic/morphological realization] pairs) to find the most similar or same substructures and the relevant syntactic relations to generate a set of syntactic realizations, which may be sentences or sentence fragments. In our current implementation, the SAGA subgraph matching algorithm [22] has been used to match the subnets from the parsed corpus at this step.

If Step 3 generated multiple fragments, they must be pieced together. In Step 4, the Link Parser's dictionary has been used for detecting the dangling syntactic links corresponding to the fragments, which can be used to integrate the multiple fragments. For instance:

In the example of Fig. 4, according to the last 3 steps, SegSim would generate two fragments: "the parser will ignore the sentence" and "whose length is too long". Then it consults the Link Parser's dictionary, and finds that "whose" has a connector "Mr-", which is used for relative clauses involving "whose", to connect to the previous noun "sentence". Analogously, we can integrate the other fragments into a whole sentence.

Finally, a "cleanup" or "post-processing" phase is conducted, applying the correct inflections to each word depending on the word properties provided by the input RelEx relations. For example, we can use the RelEx relation "DEFINITE-FLAG(cover, T)" to insert the article "the" in front of the word "cover". We have considered five factors in this version of NLGen: article, noun plural, verb intense, possessive and query type (the latter which is only for interrogative sentences).

In the "cleanup" step, we also use the chunk parser tool from OpenNLP[2] for adjusting the position of an article being inserted. For instance, consider the proto-sentence "I have big red apple." If we use the RelEx relation "noun_number(apple, singular)" to inflect the word "apple" directly, the final sentence will be "I have big red an apple", which is not well-formed. So we use the chunk parser to detect the phrase "big red apple" first, then apply the article rule in front of the noun phrase. This is a pragmatic approach which may be replaced with something more elegant and principled in later revisions of the NLGen system.

[2] http://opennlp.sourceforge.net/

3.3. Sketch of a potential connectionist SegSim implementation

We noted above our reasons for not choosing a connectionist implementation substrate for NLGen, but here we give a few brief comments regarding how one might go about creating such a substrate, if one had reason to do so.

In a connectionist approach, storage would be handled roughly as in Rohde's CSCP architecture, via training a recurrent neural network to map appropriate syntactic structures into the corresponding semantic structures. Division of target semantic structures into substructures would be done by a specialized subnetwork; and matching of each of these substructures against the knowledge base would then be done roughly as in Rohde's approach. Assembly of syntactic structures into sentence-series would then be done via a specialized subnetwork; and cleanup would be done via recurrent network trained on appropriate examples.

Overall we suspect this would be a perfectly feasible approach, but would inject computational inefficiency and additional complexity as compared to the strategy we have taken in NLGen. However, we suggest that creating something similar might become very interesting once the specific of the neural implementation of sentence generation is better understood. Right now we don't really know enough about the low-level operation of Broca's area and other relevant brain regions to create a neurologically-roughly-faithful simulation, but in a decade or two we may well.

4. NLGen: example results

NLGen is still in a relatively early stage of development, and does not handle the full range of linguistic and semantic phenomena that it will when it's completed. However, it can already express a variety of sentences encapsulating a variety of syntactic and semantic phenomena; in this section we will give some specific examples of what it can do.

The SegSim approach performs sentence generation by matching portions of propositional input to a large corpus of parsed sentences, therefore, when the successful matches can be found in the corpus, it can generate some similar and well-formed sentences via the relevant syntactic structures.

There currently do not exist any authoritative evaluation criteria for Natural Language Generation systems. Among many complicating factors here is the fact that different language generation systems have different kinds of inputs, depending on many things including their application area. So it's complicated to compare the results of NLGen with those obtained by other systems.

It is easier however to test whether NLGen is implementing SegSim successfully. One approach is to take a sentence, run it through RelEx to generate a set of relationships, and see if NLGen can regenerate the sentence from the relationship. We show here the results of some simple tests of this nature, performed with the current version of the system using a very small test corpus for similarity matching. Note: In each of these example results, the input of NLGen is the RelEx relationships produced by the sentence before "⇒"; and the sentence to the right side of "⇒" was one of the sentences generated by NLGen.

Example 1. I study mathematics at Xiamen University.
⇒ I study mathematics at Xiamen_University.

Example 2. The red block is next to the blue cup.
⇒ The red block is next_to the blue cup.

Example 3. Ruiting walked from the table to the door.
⇒ Ruiting walked to the door from the table.

Example 4. I intend to walk very slowly from Ben to Ruiting, with Hugo's pencil in my hand.
⇒ With Hugo's pencil in my hand, I intend to walk very slowly from Ben to Ruiting.

Example 5. I would like you to tell me where the red ball is.
⇒ I tell like you to 'd me where the red ball is.

Example 6. I study at Xiamen University, which is located in China.
⇒ I study at Xiamen_University.

Example 7. What is next to the tree?
⇒ What is next to the tree?

Example 8. Where is the red ball?
⇒ Where is the red ball?

Example 9. Pass me the ball.
⇒ Pass me the ball.

Example 10. Tell Ben where the red ball is.
⇒ Tell Ben where the red ball is.

To make the process clearer, for Example 1 we also show the RelEx relationships produced from the sentence before the "⇒": Fig. 5 shows the relationships of Example 1 fed to NLGen as input. The types of the semantic relationship are documented in the RelEx's wiki pages.[3]

These examples illustrate some key points about the current version of NLGen. It works well on simple, commonplace sentences (Examples 1, 2), though it may reorder the sentence fragments sometimes (Examples 3, 4). On the other hand, because of its reliance on matching against a corpus, NLGen is incapable of forming good sentences with syntactic structures not found in the corpus (Examples 5, 6). On a larger corpus these examples would have given successful results. In Examples 5, the odd error is due to the presence of too many "_subj" RelEx relationships in the relationship-set corresponding to the sentence, which distracts the matching process when it attempts to find similar substructures in the small test corpus. Then from Examples 7–10, we can see NLGen still works well for question sentences and imperative sentence if the substructures we extract can be matched, but the substructures may be similar with the assertive sentence, so we need to refine it in the "cleanup" step. For example: the substructures extracted for the sentence "are you a student?"

```
pos(., punctuation)
noun_number(Xiamen_University, singular)
pos(Xiamen_University, noun)
DEFINITE-FLAG(Xiamen_University, T)
pos(at, prep)
_subj(study, I)
at(study, Xiamen_University)
tense(study, present)
pos(study, verb)
gender(I, person)
noun_number(I, singular)
pos(I, noun)
PRONOUN-FLAG(I, T)
DEFINITE-FLAG(I, T)
```

Fig. 5. RelEx relationships for Example 1.

are the same as the ones for "you are a student?", since the two sentences both have the same binary RelEx relationships:

"_subj(be, you)
_obj(be, student)"

which are used to guide the extraction of the substructures. So we need to refine the sentence via some grammatical rules in the post-processing phase, using the word properties from RelEx, like "TRUTH-QUERY-FLAG(be, T)" which means that the referent "be" is a verb/event and the event involved is a question.

The particular shortcomings demonstrated in these examples are simple to remedy within the overall NLGen framework, via simply expanding the corpus. However, to get truly general behavior from NLGen it will be necessary to insert a fairly sophisticated rule-based generation method to cover those cases where similarity matching fails, as discussed above. The NLGen2 system created by Blake Lemoine [23] is one possibility in this regard: based on RelEx and the link parser, it carries out rule-based generation using an implementation of Chomsky's Merge operator. Integration of NLGen with NLGen2 is currently being considered. We note that the Merge operator is computationally inefficient by nature, so that it will likely never be suitable for the primary sentence generation method in a language generation system. However, pairing NLGen for generation of familiar and routine utterances with a Merge-based [24] approach for generation of complex or unfamiliar utterances, may prove a robust approach. Finally, the possible neurolinguistic analogue of NLGen2's Merge operator is a fascinating topic which we may consider in later research.

5. Conclusion

Language generation is a complex process, and at the current time we lack a thorough knowledge of either how the human brain achieves it, or how to achieve it with robust human-level functionality in software or hardware (whether brain-like in architecture or otherwise). Here we have outlined a novel approach to the problem: SegSim, which is based on a combination of local processing with massively parallel global similarity matching. As a hypothesis about human neural and cognitive function, SegSim is compatible with current brain science; and as a guide for implementation of software systems, SegSim has proved worthwhile via inspiring NLGen, a natural language generation system that, while still under development, is already yielding interesting practical results, some of which we have reported here.

[3] http://www.opencog.org/wiki/RelEx#Relations_and_Features

Acknowledgement

The authors would like to acknowledge Hugo de Garis, Changle Zhou and Xiaodong Shi at Xiamen University, for supporting Ruiting Lian in her work on this project. A debt is also owed to Karin Verspoor for her inspiration in the early design of the RelEx language engine; to Jeffrey Epstein for his support of Ben Goertzel during several phases of the research; to Kirk McMurray for his support of Linas Vepstas's work via his role at Cerego; to Borislav Iordanov for his support of Murilo Queiroz's work via his role at Dade County; and to David Stern whose investment in Novamente LLC helped support Rui Liu's work. And we are also grateful to the anonymous reviewers for their many helpful comments.

This work funded in part by Chinese National Science Foundation Grant # 60975084/F030603.

References

[1] B. Goertzel, J. Pitt, M. Ikle, C. Pennachin, R. Liu, Glocal memory: a design principle for artificial brains and minds, Neurocomputing, this issue, doi:10.1016/j.neucom.2009.10.033.
[2] B. Goertzel, D. Hart, OpenCog: a software framework for integrative AGI, in: Proceedings of AGI-08, IOS Press, 2008, pp. 468–472.
[3] C.R. Fletcher, Levels of representation in memory for discourse, in: M.A. Gernsbacher (Ed.), Handbook of Psycholinguistics, Academic Press1994, pp. 589–608.
[4] K. Bock, W. Levelt, Language production: grammatical encoding, in: M.A. Gernsbacher (Ed.), Handbook of Psycholinguistics, Academic Press1994, pp. 945–984.
[5] A. Belz, E. Kow, System building cost vs. output quality in data-to-text generation, in: Proceedings of the 12th European Workshop on Natural Language Generation (ENLG 2009), pp. 16–24.
[6] Y.W. Wong, R.J. Mooney, Generation by inverting a semantic parser that uses statistical machine translation, in: Proceedings of Human Language Technologies: The Annual Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology (HLT-NAACL 2007), pp. 172–179.
[7] P. Piwek, K. van Deemter, Constraint-based natural language generation: a survey, Technical Report 2006/03, Computing Department, The Open University, 2006.
[8] A. Koller, M. Stone, Sentence generation as a planning problem, in: Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, 2007, pp. 336–343.
[9] S. Haller, E.W. Radue, M. Erb, W. Grodd, T. Kircher, Overt sentence production in event-related fMRI, Neuropsychologia 43 (5) (2005) 807–814.
[10] N.T. Sahin, S. Pinker, E. Halgren, Abstract grammatical processing of nouns and verbs in Broca's area: evidence from fMRI, Cortex 42 (4) (2006) 540–562.
[11] C. Ranganath, M.K. Johnson, M. D'Esposito, Prefrontal activity associated with working memory and episodic long-term memory, Neuropsychologia 41 (3) (2003) 378–389.
[12] C.J. Fiebach, M. Schlesewsky, G. Lohmann, D.Y. von Cramon, A.D. Friederici, Revisiting the role of Broca's area in sentence processing: syntactic integration versus syntactic working memory, Human Brain Mapping 24 (2005) 79–91.
[13] C.J. Fiebach, R.I. Schubotz, Dynamic anticipatory processing of hierachical sequential events: a common role for Broca's area and ventral premotor cortex across domains, Cortex 42 (2006) 499–502.
[14] J. Kalita, L. Shastri, Generation of simple sentences in english using the connectionist model of computation, in: Proceedings of the 9th annual conference of the Cognitive Science Society (Hillsdale, NJ: Lawrence Erlbaum Associates, 1987, pp. 555–565.
[15] J. Kalita, L. Shastri, A connectionist approach to generation of simple sentences and word choice, in: G. Adriaens, U. Hahn (Eds.), Parallel Natural Language Processing, Ablex Publishing, Norwood, NJ1994, pp. 395–420.
[16] M.E. Gasser, A connectionist model of sentence generation in a first and second language, Unpublished doctoral dissertation, Computer Science Department, University of California, Los Angeles, CA. (TP UCLA-AI-88-13), 1988.
[17] R. Miikkulainen, in: Subsymbolic Natural Language Processing: An Integrated Model of Scripts, Lexicon, and Memory, MIT Press, 1993.
[18] D.L.T. Rohde, A connectionist model of sentence comprehension and production, Ph.D. thesis, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, 2002.
[19] D.J. Amit, in: Modeling Brain Function—The World of Attractor Neural Networks, Cambridge University Press, New York, USA, 1989.
[20] B. Goertzel, H. Pinto, A. Heljakka, M. Ross, I. Goertzel, C. Pennachin Using dependency parsing and probabilistic inference to extract gene/protein interactions implicit in the combination of multiple biomedical research abstracts, in: Proceedings of the BioNLP—2006 Workshop at ACL-2006, (New York, 2006), pp. 104–111.
[21] D. Sleator, D. Temperley, Parsing English with a link grammar, in: Proceedings of the Third International Workshop on Parsing Technologies, 1993.
[22] Y. Tian, R.C. Mceachin, C. Santos, D.J. States, J.M. Patel, SAGA: a subgraph matching tool for biological graphs, Bioinformatics 23 (2) (2007) 232–239.
[23] B. Lemoine, NLGen2: a linguistically plausible, general purpose natural language generation system, 〈http://www.louisiana.edu/~bal2277/NLGen2.doc〉.
[24] Noam Chomsky, in: R. Martin, D. Michaels, J. Uriagereka (Eds.), Minimalist Inquiries: The Framework. In Step by Step, MIT Press, Cambridge2000, pp. 89–155.
[25] R. Granger Richard, Engines of the brain: the computational instruction set of human cognition, AI Magazine 27 (2006) 15–32.

Ruiting Lian is a Ph.D. student in Artificial Brain Lab, Xiamen University, China. Research interests: Natural Language Processing and Artificial Brain.

Dr. Ben Goertzel is a CEO and a Chief Scientist of AI firm Novamente LLC, a company focused on creating powerfully intelligent NPC's for online games and virtual worlds. He is also a CEO of bioinformatics firm Biomind LLC, and a Director of Research of the nonprofit Singularity Institute for AI. Dr. Goertzel is the originator of the OpenCog open-source AGI framework, as well as the proprietary Novamente Cognition Engine AGI system. A research faculty for 8 years in several universities in the US and Australasia, he remains active in the academic AI community. He is the Chair of the Steering Committee for the Artificial General Intelligence conference series, and was the Conference Chair for AGI-09 which was held in March 2009 in Washington DC. He currently serves on the Board of the World Transhumanist Association. Dr. Goertzel has authored eight technical monographs in the computing and cognitive sciences, published by leading scientific publishers, most recently Probabilistic Term Logic, published by Springer in 2008; and also edited four technical volumes. He has also published over 80 research papers in journals, conferences and edited volumes, in disciplines spanning AI, mathematics, computer science, cognitive science, philosophy of mind and bioinformatics; and has developed two AI-based trading systems for hedge funds in Connecticut and San Francisco. AI software created by his teams at Novamente LLC and Biomind LLC has been used in numerous government agencies and corporations. Based in Rockville Maryland, he is married with three children aged 19, 16 and 12; and in his spare time he writes avant-garde fiction and composes and improvises experimental keyboard music.

Rui Liu is a Ph.D. student at Wuhan University in China, with multiple areas of interest and expertise including evolutionary learning, evolvable hardware, and financial prediction. He is also an experienced software entrepreneur and commercial software developer, and in 2008–2009 did AI software development for Novamente LLC.

Michael Ross has a degree in Symbolic Systems from Stanford University. He is an independent computational linguistics consultant for clients in the financial, defense, and health industries. While working for SAIC, he lead design and development of the RelEx system, now part of OpenCog. He lives in New York City, and occasionally contributes to Lingpipe, an open source natural language processing suite.

Murilo Saraiva de Queiroz, 32, has a degree in Computer Science and a Masters Degree in Electronics Engineering, both from Universidade Federal de Minas Gerais (UFMG, Brazil). He is a partner in Vetta Labs (http://www.vettalabs.com), a research and development company focused on creating innovative products in areas such as financial markets, bioinformatics, computer vision and natural language processing. He lives in Belo Horizonte, Brazil, with his wife and son.

Dr. Linas Vepstas received a Ph.D. in theoretical physics from SUNY at StonyBrook. Linas has had a long career computer engineering industry, working on many areas in software, operating systems and hardware, strongly supporting open source development. He currently pursues research into mathematics linguistics, and theories of unsupervised learning in artificial intelligence. When not engaged in intellectual pursuits, he competes in rowing regattas in his home town of Austin, Texas.