EasyChair Preprint

№ 6171

# Parsing Text in a Workspace for Language Generation

George Wright and Matthew Purver



July 27, 2021

# Parsing Text in a Workspace for Language Generation

George Wright[1] and Matthew Purver[1]

[1]Cognitive Science Research Group, School of Electronic Engineering and Computer Science, Queen Mary University of London

## Author Note

The authors declare that there no conflicts of interest with respect to this preprint.

Correspondence should be addressed to First author name and address. Email: george.a.wright@qmul.ac.uk

**Abstract**

Processing of language by humans involves the intertwining of processes of production and comprehension. This paper describes how a cognitively inspired architecture for analogy-making can be adapted for the modeling of language generation and specifically details how a generated sentence can be parsed within a workspace in order to contribute to the program's self-monitoring and self-evaluation.

*Keywords:* workspace, codelet, natural language generation, parsing

**Parsing Text in a Workspace for Language Generation**

This research aims to simulate human creativity in generating language. It adopts a cognitively inspired workspace-based architecture in which production and comprehension can interact so that self-monitoring and self-evaluation can co-occur with and influence text generation. This paper describes how sentence parsing can take place in such an architecture and how this can help text generation.

The computer program works in the domain of weather description. This provides a test-domain which is conceptually simple – only limited knowledge is required – but still linguistically challenging – information contained in many dimensions (a 2-dimensional map, multiple aspects of weather, time) must be selected and arranged into a linear text. This early iteration of the computer program works only with temperatures on a 2-dimensional map.

The architecture of the program is based on work by the Fluid Analogies Research Group such as Copycat. Copycat (Mitchell, 1993) is a model of analogy making which completes analogies between strings of the form ABC:ABD::IJK:?. Spreading activation in its concept network influences the selection of micro-agents called *codelets* which build structures in a workspace in order to solve the problem. For example, a codelet which recognizes that B is the SUCCESSOR of A will cause the SUCCESSOR concept to become more active and therefore encourage top-down codelets to seek out more examples of the SUCCESSOR relation and eventually complete the analogy accordingly. The program's lack of centralized control and stochasticity allow it to simultaneously consider multiple pathways to different solutions. Less promising pathways are gradually abandoned as a result of competition between structures in a search strategy called a *parallel terraced scan*. The program uses *computational temperature* – a

measure of the quality and coherence of the workspace – to determine how random codelet selection should be. As pathways are narrowed down, processing becomes more deterministic.

Numbo (Defays, 1995) is a related program which plays a number game in which a target number is made out of smaller numbers using addition, subtraction, and multiplication. For example, when given the target 114 and the numbers 11, 20, 7, 1, and 6, possible solutions include:

$$20 \times 6 - 7 + 1$$

$$(20 - 1) \times 6$$

Like Copycat, Numbo's permanent knowledge contains concepts which influence codelet activity in the workspace as they become activated. But, it also contains structured information in the form of *bipeds* which encode declarative knowledge of operations on landmark integers such as $6 = 2 \times 3$ and $100 = 5 \times 20$. Analogy-making between prototypical operations represented in the concept network's bipeds and numbers in the workspace guides the search for a solution.

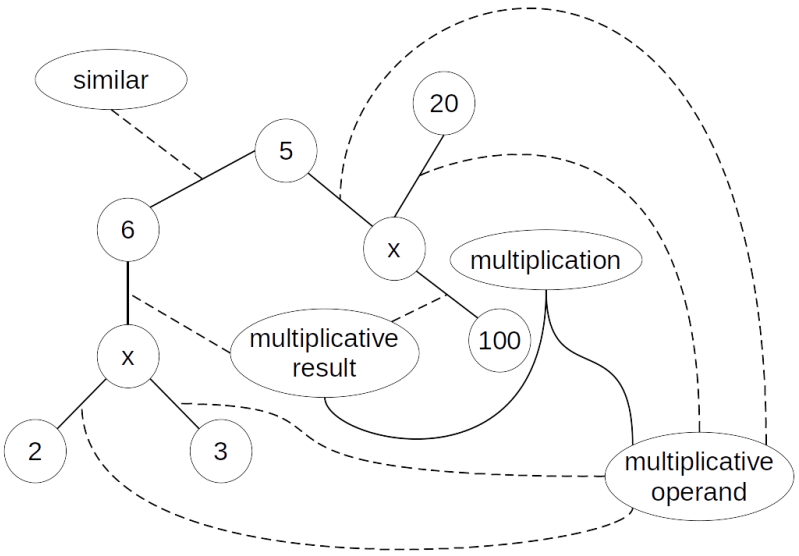


*Figure 1: Part of Numbo's conceptual network with two bipeds for multiplication (Defays, 1995, p.136).*

Because these architectures center on a workspace where many processes take place concurrently, they are also ideal for modeling the interleaving of production and comprehension processes which occur when humans process language (Pickering & Garrod, 2013). They should also allow for interference between production and comprehension since residual activation of a concept as a result of comprehension would make it more likely to influence production.

Gan *et al*. (1996) show how codelets operating at the level of the sentence, phrase, and word can solve the problem of ambiguous word boundaries in Chinese. But there has been little further work using this style of architecture in language processing.

## Method

The architecture centres around four components:

1. A collection of workspaces where the input, intermediate structures, and output text are worked on.
2. A collection of conceptual spaces where domain-specific concepts such as HOT and grammatical concepts such as NOUN are stored. Not all concepts are connected as part of a network as in Copycat and Numbo: temperature and location concepts are stored respectively in a TEMPERATURE and LOCATION space where a distance metric rather than explicitly instantiated connections determine similarity between concepts. The conceptual spaces therefore sit between the vector space representations described by Gärdenfors (2014) and more traditional symbolic networks.
3. The coderack, where codelets wait to be selected stochastically to enter the workspace. Each codelet is responsible for evaluating or altering workspace structures.
4. A measure of the model's satisfaction with its work so far, equivalent to 1 - *temperature* in the aforementioned programs, here called *satisfaction* to avoid confusion with weather. Lower satisfaction results in more random codelet selection so that more diverse alternatives can be explored.

Structures built by the program include *chunks* used to recognize homogeneous regions on a map; *labels* which classify items, for example a chunk could be labeled HOT and the word “hot” labeled ADJECTIVE; *relations* between two items, for example one chunk may be MORE hot than another; *correspondences* which indicate that two items, for example part of the input and an element in a frame or template are the SAME; correspondences between elements of the input

and template slots allow for the generation of *words*; parsing of words results in the creation of *phrases* which are chunks of words labeled with a grammatical role such as NOUN-PHRASE.

The program starts with a workspace containing a 2-dimensional map of the weather. As the program runs, label, chunk, and relation building codelets divide the map into areas of similar weather, classify the temperatures as COLD, WARM, *etc* and make comparisons between the temperatures. Templates containing common weather description phrases help sentence construction: correspondences connect relevant label values to relevant template slots and words are placed into an output sentence. Phrase building codelets check that the sentence is complete and an interpretation of chunks and labels is reverse-engineered out of the text. If correspondences can be built connecting the interpretation to the original input, the text is deemed adequate and is good enough to be output. Throughout the program's run multiple structures which are not necessarily consistent with one another can be built. If correspondences cannot be built between an output and the input, alternative structures and textual outputs must be selected for. This architecture makes language understanding an important and integrated part of the generative process whereas natural language generating programs have traditionally been unidirectional and modular (Gatt & Krahmer, 2018, p.82-101).

**The Program as Parser**

From the program's codelets emerge macro-processes for data interpretation, language generation, and language understanding.

The (constituency) parsing components of this program are analogous to Numbo: bipeds containing mathematical operations are replaced with bipeds containing grammatical rules. Codelets gradually build a parse tree by labeling words with part-of-speech tags and chunking them together into larger phrases if they match with a rule.

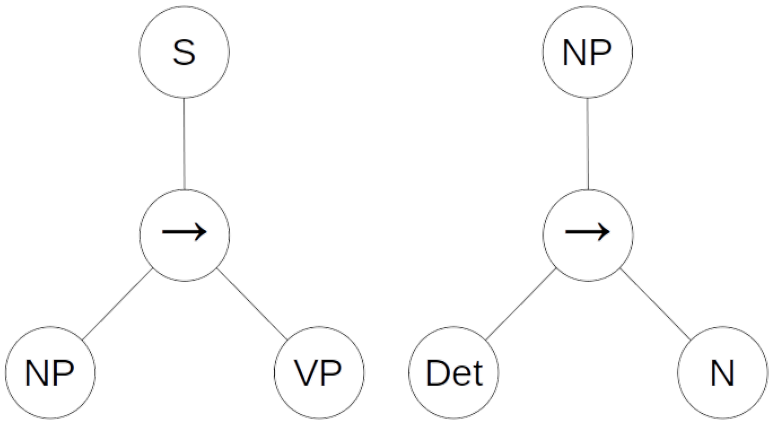


*Figure 2: Numbo-style bipeds for a context-free grammar.*

This bears some resemblance to chart parsing: the workspace is essentially a chart of intermediate structures. Label and phrase evaluation and selection codelets decide which of the structures receive further attention and which are abandoned.

Evaluation codelets determine the quality of structures. Label quality is determined by the likelihood that the word is an instance of the label. Phrase quality is determined by:

- The quality of the constituent branches,
- The activation of the rule,
- The number of phrases it is contained within (how useful the phrase turned out to be in further parsing),

- The length of the phrase (this prevents low quality scores for phrases high up the parse tree).

Selection codelets choose between two competing structures, in this case alternative structures which cannot both belong in the same parse tree. Selection codelets select two competing structures and probabilistically boost the activation of the higher quality structure and dampen the activation of the lower quality structure. Lower quality structures still have some chance of being selected in case they can be used to create a better overall parse.

## Results

When treating the program as nothing but a parser using a context free grammar, out of 1000 runs, it took on average 613 codelets to parse the sentence “it is warm in the south”. The program's non-determinism allows for it to use left-recursive rules such as $s \rightarrow s, pp$.

In one example run, label building codelets first apply labels to words and then gradually phrase building codelets try to construct phrases. They are unable to do so until both words in a potential phrase have part-of-speech labels compatible with a rule. For example, “the” and “south” are at first labeled DET and ADJ respectively but a noun-phrase can only be created once “south” has been labeled with NOUN. As with the model in Gan *et al* (1996, p.547), processing tends to move from lower level units (words) to higher level units (phrases), but the ordering is not strict and can be interleaved.

Currently the program uses a bottom-up strategy of randomly classifying and pairing up structures to try and make phrases. Better use of the spreading activation network could improve the efficiency of the search. For example, the DET concept could spread activation to the NOUN-PHRASE concept to push the search in a more fruitful direction resulting in something more like a left-corner parser.

## Discussion

Parsing contributes to the model's language comprehension abilities, allowing it to know when a sentence is correct and complete. This should give it the ability to produce language more fluid than language based on templates alone.

For example, it could cut short a sentence such as “it is warmer in the south than the north” to “it is warmer in the south” or “it is warmer” when context allows. A better understanding of the grammar of sentences should also allow for better text manipulation when combining multiple sentences, for example deletion of repeated subjects.

Of course, for the grammatical knowledge to be made use of, it needs to exist alongside other levels of processing including at the level of semantics and discourse. Future iterations of this program must include codelets operating at these levels.

Whether or not grammatical knowledge improves the output of the program can be tested by comparing the program's behaviour with and without parsing enabled. Multiple runs of each version of the program will show the distribution of outputs it can produce as well as the length of time (or number of codelets) required to produce an answer. The quality of the outputs can also be compared by human judges.

### Comparison with Related Work

Similar work which makes use of parsing or comprehension for language production include cognitive models such as that proposed by Pickering and Garrod (2013) and work which makes use of the dynamic syntax paradigm such as Purver and Otsuka (2003).

The architecture described above bears some resemblances to that described by Pickering and Garrod with templates standing in for their forward models (impoverished, easy to compute representations) and the correspondences built between input and templates matching the comparison between the output of the production implementer and the forward model. The use of parsing for self-comprehension, however, is more similar to the internal loop of more traditional models such as Wheeldon and Levelt (1995), which Pickering and Garrod do not discount as also playing a role in self-monitoring albeit at a different level (Pickering & Garrod, 2013, p.340).

The architecture in its current form does not match well with models based on dynamic syntax since in these models, generation and parsing are one and the same process as opposed to two interleaved processes. That said, it may be worth considering dynamic syntax or other formalisms as an alternative to context-free grammar. Dynamic syntax does not prescribe a particular algorithm or architecture and since it is also a representation based on nodes and links, this architecture of codelets incrementally building structures in a workspace can be readily adapted to it.

## Acknowledgments

Purver is partially supported by the EPSRC under grant EP/S033564/1, and by the European Union's Horizon 2020 program under grant agreement 825153 (EMBEDDIA, Cross-Lingual Embeddings for Less- Represented Languages in European News Media). The results of this publication reflect only the authors' views and the Commission is not responsible for any use that may be made of the information it contains.

## References

Defays, D. (1995). Numbo: A Study in Cognition and Recognition. In Hofstadter D. J. (Ed.) *Fluid Concepts and Creative Analogies*. (pp. 131-154). Basic Books.

Gan, K. W., Lua, K. T., & Palmer, M. (1996). A Statistically Emergent Approach for Language Processing: Application to Modeling Context Effects in Ambiguous Chinese Word Boundary Perception. *Computational Linguistics*, 22(4), 531-553.

Gärdenfors, P. (2014). *The Geometry of Meaning: Semantics Based on Conceptual Spaces*. The MIT Press.

Gatt, A., & Krahmer, E. (2018). Survey of the State of the Art in Natural Language Generation: Core Tasks, Applications, and Evaluation. J*ournal of Artificial Intelligence Research*, 61, 65-170.

Mitchell, M. (1993). *Analogy-Making as Perception: A Computer Model*. The MIT Press.

Pickering, M. J., & Garrod, S. (2013). An Integrated Theory of Language Production and Comprehension. *Behavioral and Brain Sciences*, 36, 329-392.

Purver, M. and Otsuka, M. (2003). Incremental Generation by Incremental Parsing: Tactical Generation in Dynamic Syntax. In *Proceedings of the 9th European Workshop on Natural Language Generation.* (pp. 79-86).

Wheeldon, L. R., & Levelt, W. J. M. (1995). Monitoring the Time Course of Phonological Encoding. *Journal of Memory and Language*, 34(3), 311-334.