Developing multi-level grammars in a framework of information compression by multiple alignment, unification and search

Gerry Wolff
CognitionResearch.org.uk
Menai Bridge, UK.
jgw@cognitionresearch.org.uk

Abstract

This paper is a progress report on recent work developing the SP framework for computing and cognition to perform unsupervised grammatical inference. The SP70 computer model, presented in previous publications, is briefly described, highlighting the main weakness of the model, that, while it can discover structure at the levels of sentences and words, it is not able to discover intermediate levels of structure such as phrases or clauses.

The main body of this paper describes SP71, a successor to SP70, that largely overcomes the weakness of SP70. While there is still work to be done in refining the model, it can abstract intermediate levels of structure from appropriate input and it can create plausible grammars for those kinds of raw data.

The body of SP71 is an iteration of two phases, one phase that derives new grammatical patterns from parsings of the input and a second phase that compiles grammars from these patterns that are 'good' in terms of principles of Minimum Length Encoding and discards structures that are not proving useful in those terms. These two phases are organised so that intermediate levels of structure may be abstracted.

1 Introduction

This paper is a progress report on recent work developing the SP theory of computing and cognition to accommodate grammatical inference, aiming to overcome a particular weakness an earlier model which will be described.

The SP theory, which has been under development since 1987, aims to integrate a wide range of concepts in computing and cognition within one relatively simple conceptual framework. It is founded on principles of Minimum Length Encoding pioneered by [Solomonoff, 1964; Wallace and Boulton, 1968; Rissanen, 1978] and it incorporates a complimentary idea with a long history in cognitive psychology: that many aspects of the workings of brains and nervous systems may be understood as information compression.

The theory is conceived as an abstract model that receives *New* information from its environment and transfers this information to a repository of *Old* information. At the same time, it is designed to compress the New information as much as possible by searching for patterns or parts of patterns that are the same and merging or *unifying* those matching patterns or parts of patterns. An important part of this process is the building of *multiple alignments*, similar to multiple alignments in bioinformatics but with important differences. An overview of this programme of research may be found in [Wolff, 2003a] and earlier papers that are cited therein.

1.1 The SP Theory and Unsupervised Grammatical Inference

In current SP models, all knowledge in the system is encoded as a set of one-dimensional sequences or *patterns* and these may be structured to be formally equivalent to a context-free or context-sensitive grammar, as described in [Wolff, 2000].[1] Thus learning in the system may be cast as grammatical inference.

Although, at an abstract level, the SP framework is a model for learning, it is not until relatively recently that this aspect of the framework has been addressed in detail. In what follows, I shall briefly describe the SP70 computer model—a first model of grammatical inference within the SP framework—and then in more detail I shall describe the SP71 model, designed to overcome a significant weakness in the SP70 model.

2 The SP70 Model and Its Strengths and Limitations

The SP70 computer model, described in [Wolff, 2003b; 2002], receives a set of sentences as input and derives one or more alternative sets of patterns, each set representing a grammar for the original sentences. Each of these alternative grammars has a measure of 'goodness' in terms of principles of Minimum Length Encoding.

A key point to note is that the original sentences are presented as a sequence of letters without any kind of spacing or other marker between one word and the next. Thus, although the system does not have any concept of 'word', it must discover or infer which subsequences of the original sentences are significant and would be recognised by people as words.

[1] It is envisaged that, in future development of the framework, the concept of *pattern* may be generalised to two-dimensional arrays of symbols but this has not yet been attempted.

It must also discover, for each of several different syntactic contexts, groupings of words that are distributionally equivalent in the given context and it must discover or construct an abstract pattern that records the sequence of words and disjunctive groupings of words. This abstract pattern is equivalent to the rule in a conventional grammar that represents the concept of 'sentence'.

The development of this model was a major step forward and corroborated the anticipated value of principles of Minimum Length Coding in the inference of plausible grammars from raw input without the kinds of 'supervision' described by [Gold, 1967] and without explicit marking of word boundaries within sentences and without information about distributional groupings of words. However, a significant weakness of the SP70 model is that it is only able to infer structures at two levels of abstraction: words and sentences. It is not capable of inferring intermediate levels of structure such as 'phrases' or 'clauses' and this weakness is inherent in the design of the model, not merely an accidental feature of the test materials that have been used.

Although the SP framework can accommodate context-sensitive grammars (as described in [Wolff, 2000]), no attempt has yet been made to learn grammars of this type. All the work to date, including work described in this paper, has concentrated on the learning of context-free phrase-structure grammars.

3 The SP71 Model: Learning Grammars with Intermediate Levels of Structure

The SP71 model is intended to learn grammars with intermediate levels of structure. Although there is still work to be done in refining the model, it can already demonstrate how such grammars may be learned within the SP framework. The description of the model in this section will explain the overall organisation of the model and key features of how it works. After a more detailed description of the model in the following sections, Section 7, below, explains how the model learns intermediate levels of structure.

The basic elements of the model are similar to SP70 but some key changes have been made in some of those basic elements and the overall structure of the program—shown in Figure 1—has been changed quite radically. Initially, the New patterns (representing the sentences for which one or more grammars are required) are read in to the program and from these patterns the program finds the frequency of occurrence of each symbol type. From these frequencies, it calculates the number of bits of storage to be associated with each symbol using the Shannon-Fano-Elias method (see [Cover and Thomas, 1991]). These *bit costs* are used in the evaluation of multiple alignments and in the evaluation of grammars that are compiled by the program.

There are two main phases in the main body of the model: the first one generates patterns to be added to Old and the second one selects amongst those many patterns by generating alternative grammars and replacing all the Old patterns with the patterns in the best N grammars, where N is normally about 3. These two phases are applied to each New pattern in succession until all the New patterns have been processed.

4 Phase 1: the Derivation of Patterns to be Added to Old

As in SP70, each New pattern is 'parsed' by the formation of *multiple alignments* between the New pattern and one or more Old patterns (step P1.2 in Figure 1). In brief, a multiple alignment in this context is an arrangement of patterns that shows matching of symbols between patterns, with judicious 'stretching' of patterns where necessary. And the system is designed to seek out multiple alignments that allow the New pattern to be encoded economically in terms of the Old patterns, in accordance with principles of Minimum Length Encoding (see [Wolff, 2000]). In broad terms, this means maximising the amount of matching between patterns.

If an alignment is 'full', meaning that all relevant symbols are matched, then it is no use for the derivation of new patterns. What is needed in Phase 1 are alignments that are partial, meaning that there are unmatched symbols in the New pattern, or one of the Old patterns, or both.

4.1 A Simple Example

In simple cases, patterns are derived from multiple alignments in almost exactly the same way as in SP70. From an alignment like the one shown in Figure 2, the system creates patterns like those shown in Figure 3.

```
0      w e w a l k f a s t   0
       | |         | | | |
1 < #1 w e r u n   f a s t > 1
```

Figure 2: A simple alignment created by SP71. By convention in all alignments, the New pattern is shown in row 0 and Old patterns are shown in other rows, one pattern per row.

```
< 13 w e >
< 14 #17 r u n >
< 14 #18 w a l k >
< 15 f a s t >
< #19 < 13 > < 14 > < 15 > >
```

Figure 3: Patterns that the system derives from the alignment shown in Figure 2

In deriving patterns from an alignment, the system looks for coherent sequences of matched symbols or coherent sequences of unmatched symbols, always excluding 'identification' symbols (ID-symbols) like '$<$ #1 ... $>$' in row 1 of Figure 2. The system adds new ID-symbols to patterns that it has derived—as can be seen in the first four patterns in Figure 3. And it uses copies of those ID-symbols as references within the fifth pattern in the same figure. This last pattern is the 'abstract' pattern mentioned earlier that describes the overall structure of the original sentences and corresponds to the 'sentence' rule in a conventional grammar.

Notice that 'r u n' and 'w a l k' have both been given the ID-symbol '14' and a copy of that symbol is also used in the middle of the fifth pattern in the figure, showing that 'r u n' and 'w a l k' are, for syntactic purposes, interchangeable in that position in the sentence. This is the beginning of a grammatical class like 'verb' in a fully-developed grammar.

```
SP71()
{
     1 Read a set of patterns into New. Each New pattern is normally a sentence
          expressed as a sequence of letters without spaces or punctuation
          between words. Old is initially empty.
     2 Make a preliminary measure of the frequencies of occurrence of the symbol
          types in the New patterns. From these frequencies, derive a bit cost
          for each symbol type using the Shannon-Fano-Elias method.
     3 While (there are unprocessed patterns in New)
     {
          PHASE 1: DERIVE PATTERNS FROM PARTIAL ALIGNMENTS.
               P1.1 Identify the first or next pattern from New as the 'current'
                    New pattern (CNP).
               P1.2 Parse the CNP by multiple alignment using the patterns that
                    are currently in Old.
               P1.3 Derive new patterns from those multiple alignments that are
                    'partial', as explained in the text.
               P1.4 Add the derived patterns to Old together with a copy of the
                    CNP to which 'identification' symbols have been added (ID-CNP).
          PHASE 2: COMPILE ALTERNATIVE GRAMMARS.
               P2.1 Reparse all the New patterns up to and including the CNP.
               P2.2 From all the multiple alignments that are formed, select a subset
                    that are 'full' alignments, as explained in the text.
               P2.3 From the full alignments, derive frequencies of Old patterns,
                    frequencies of symbol types, and bit costs for symbol types.
               P2.4 From the full alignments, compile a set of alternative grammars for
                    all the New patterns up and including the CNP, as explained in
                    the text.
               P2.5 Delete all the Old patterns and replace them with the patterns in
                    the best N grammars that have been found (where N is typically
                    about 3), together with all ID-CNPs up to and including the
                    current one. Clean unnecessary ID-symbols from this set of Old
                    patterns.
               P2.6 Delete all grammars ready for the next iteration, except when the
                    CNP is the last New pattern.
     }
     4 Clean and tidy the best three grammars and print.
}
```

Figure 1: The organisation of SP71. Key parts of the process are explained in the text.

The symbols '#17' and '#18' in '< 14 #17 r u n >' and '< 14 #18 w a l k >' are 'discrimination' ID-symbols that allow the two patterns to be distinguished from each other by their ID-symbols.

With regard to the pattern '< #19 < 13 > < 14 > < 15 > >', the symbols '< #19 ... >' are ID-symbols for the pattern whereas the symbols '< 13 > < 14 > < 15 >' represent the substance or 'contents' of the pattern and are designated C-symbols. In general, ID-symbols are normally at the beginning and end of each pattern while C-symbols are the remaining symbols within the body of the pattern.

A point to notice about the derivation of new patterns is that ID-symbols and copies of them in the abstract pattern are newly-created by the system. Consequently, they do not at this stage have any value for their frequency of occurrence or bit cost from step P2.3 of the framework shown in Figure 1. As a temporary measure, an approximate value is assigned to them. However, when step P2.3 is reached, these approximate values are replaced by much more precise values derived from the patterns and symbols that are actually in Old at that time.

4.2 Deriving Patterns from Alignments Containing Three or More Patterns

Figure 4 shows a slightly more complicated alignment in which 'y o u' in row 0 and '< 13 >' in row 3 are not matched to anything. As before, we ignore the ID-symbols in the Old pattern which, in this case, are the symbols '< #19 ... >' in the pattern in row 3.

The alignment shown in Figure 4 raises the general question "How does the system deal with alignments that contain more than two patterns?" The answer depends on an understanding of how alignments are formed. In SP71, alignments are built in a step-wise manner, adding one row at a time, always at the bottom. This is different from all earlier SP models where alignments may be built by combining pairs of alignments, either of which may contain two or more patterns. Because alignments are built in this step-wise manner in SP71, it is possible to enforce the following rule: "An alignment can only become part of a larger alignment if all its C-symbols are matched." As soon as an alignment is formed in which some of its C-symbols are unmatched, then it cannot be part of any larger alignment. This means that in any partial alignment, of any complexity, there are only two patterns to

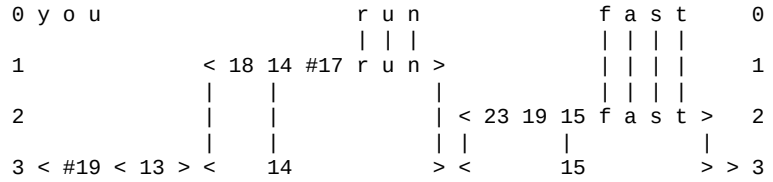

Figure 4: A four-row alignment from which new patterns are derived.

be considered: the New pattern (which by convention is always shown in row 0) and the Old pattern in the last row to be added, which is always the bottom row of the alignment.

With the alignment shown in Figure 4, the system isolates the unmatched symbols 'y o u' from the New pattern and the unmatched symbols '$<$ 13 $>$' from row 3 (ignoring the ID-symbols '$<$ #19 ... $>$'). Rather than assigning new ID-symbols to both patterns, the system recognises that '$<$ 13 $>$' is already a 'reference' to a pattern or class of patterns and it assigns the ID-symbols '$<$ 13 ... $>$' to 'y o u', which converts it into '$<$ 13 y o u $>$'. In short, 'y o u' is assigned to the same syntactic class as '$<$ 13 w e $>$' from Figure 3. Discrimination ID-symbols are also added.

The system does not need to do anything about the matched sequences 'r u n' and 'f a s t' because these are already represented by Old patterns with appropriate ID-symbols.

4.3 Untidiness in the Derivation of Patterns

The examples presented so far may have given the impression that the process of deriving new patterns always accords exactly with one's intuitions about what would be appropriate for a given set of patterns. In fact, the process is very untidy and leads to the creation of many patterns which are very different from what one might think was 'correct' for a given set of New patterns. For example, an alignment like the one shown in Figure 5 leads to the creation of patterns like 'w', 'e', 'r u n f', 'a', 'l k f a' and 's t', each with added ID-symbols. There is also an abstract pattern to tie them all together.

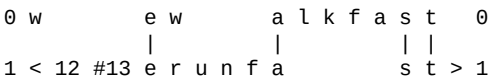

Figure 5: An 'untidy' alignment leading to the creation of counter-intuitive patterns.

Another source of untidiness is that, as learning proceeds, many patterns acquire a diverse range of ID-symbols. For example, a pattern that starts out as 'w e' finishes as '$<$ 150 147 91 85 13 #23 w e $>$' and the pattern 's l o w l y' eventually becomes '$<$ 149 131 87 #74 s l o w l y $>$'. Other less extreme examples can be seen in rows 1 and 2 of Figure 4.

The reason for this untidiness is that any given pattern may be derived from several different alignments. Since, during the learning process, it is not possible to know what the final 'best' result will be, an ID-symbol is added to a given pattern for each of the several different alignments (contexts) from which it was derived.

As we shall see, the system is able, eventually, to find intuitively plausible grammars despite these two kinds of untidiness that are so prominent during Phase 1 of the learning process.

5 Phase 2: the Compiling of Alternative Grammars

In Phase 2, for every 'current' pattern from New (CNP), the program reparses every New pattern up to and including the CNP and then it selects all the 'full' alignments, meaning alignments in which there is a match for every New symbol and every C-symbol in every Old pattern in the alignment. From these full alignments, it compiles a set of 'good' grammars for the New patterns up and including the CNP using methods outlined in items P2.3 and P2.4 of Figure 1 that are the same as the methods used in SP70.

The process of compiling good grammars is essentially a hill-climbing search through the abstract space of alternative grammars, trying to minimise $(G + E)$ for each grammar, where G is the size of the given grammar (in bits) and E is the size of all the New patterns (in bits) after they have been encoded in terms of the grammar. Minimising $(G + E)$ is, of course, the central idea in grammar induction using principles of Minimum Length Encoding.

The grammars are built in stages, at first trying to minimise $(G + E)$ for the first New pattern alone, then trying to minimise $(G + E)$ for the first and second New pattern, followed by the first, second and third, and so on.

6 Cleaning and Tidying of ID-Symbols

As was described in Section 4.3, the second source of untidiness in the learning process is the way any given pattern can acquire many different ID-symbols derived from different alignments.

In any one grammar, many of these ID-symbols will be irrelevant to the structure of the grammar. So any grammar can be 'cleaned' by removing ID-symbols that have no function in the grammar, meaning that it is never used as a reference from one pattern to another or it is not needed as a top-level reference for a pattern. SP71 contains a procedure for cleaning any grammar when that is required.

After a grammar has been cleaned, many of the numbers used as the names of ID-symbols will be unnecessarily high. From a functional standpoint, this does not matter but from a cosmetic point of view it looks rather odd. So any grammar that has been cleaned may also be 'tidied' by renumbering the ID-symbols, starting from 1.

In SP71, the set of Old patterns that is compiled in step P2.5 of Figure 1 is cleaned to remove unnecessary ID-symbols before the next iteration of step 3 of the program. The ID-symbols are not renumbered at this stage because it is necessary to maintain consistency across iterations of step 3.

Right at the end of the program (step 4 in Figure 1), the best three grammars are cleaned and the remaining ID-symbols are renumbered so that the final grammars look presentable.

7 The Creation of Grammars with Intermediate Levels of Structure

The principles of Minimum Length Encoding dictate that a structure should be recognised within a grammar if it is 'useful', meaning that it helps to minimise $(G + E)$. As a rule of thumb, structures are useful if they occur relatively frequently and if they represent a relatively large amount of the raw data.

These principles should apply if one sentence pattern is used within another. For example, 'w e r u n' can occur as a free-standing sentence and it can also occur as part of a sentence like 'w e r u n f a s t'. Given a set of sentences that contains both kinds of sentence, principles of Minimum Length Encoding should lead to the creation of grammars in which the abstract pattern that describes the short sentences would be referenced within the abstract pattern that describes the longer sentences. A grammar of that sort would contain an intermediate levels of structure of a kind that is beyond the scope of SP70.

The SP71 model is not yet fully robust in all situations but it does have the main elements needed to recognise the kind of intermediate structure just described. In what follows, a selection of results are shown to illustrate what it can do. The kind of input used is the set of patterns shown in Figure 6, or the same input with the short sentences coming before the long ones.

```
w e r u n f a s t
w e r u n s l o w l y
w e w a l k f a s t
w e w a l k s l o w l y
t h e y r u n f a s t
t h e y r u n s l o w l y
t h e y w a l k f a s t
t h e y w a l k s l o w l y
w e r u n
w e w a l k
t h e y r u n
t h e y w a l k
```

Figure 6: A set of New patterns for SP71 containing longer sentences and shorter sentences (at the end) which can be seen to occur within the longer sentences.

Given that the longer sentences come before the shorter ones, the program forms alignments like the one shown in Figure 7. In this alignment, the abstract pattern in row 3 is derived from the longer sentences that the program has seen. The New pattern in the alignment is one of the shorter sentences. From this alignment, the program derives the patterns '$<$ 145 #182 $<$ 13 $>$ $<$ 14 $>$ $>$' and '$<$ #183 $<$ 145 $>$ $<$ 15 $>$ $>$'. The first one is derived from those C-symbols in row 3 that have been matched and it describes the structure of the short sentence in row 0. This first pattern has been marked with the ID-symbol '145' and a copy of that symbol appears within the second pattern. In effect, there is a reference to the first pattern within the second pattern and this reference is followed by '$<$ 15 $>$' which is derived from the unmatched C-symbols in row 3 of the alignment. Thus the second pattern, '$<$ #183 $<$ 145 $>$ $<$ 15 $>$ $>$', describes the structure of the longer sentences by means of a reference to the shorter sentence type within it. This is the kind of intermediate level of structure and hierarchical relationship that is beyond the scope of SP70.

```
0               w e             r u n           0
                | |             | | |
1       < 13 #22 w e >          | | |           1
        | |       |             | | |
2       | |       | < 77 14 #23 r u n >         2
        | |       | |   |           |           |
3 < #25 < 13      > <   14          > < 15 > > 3
```

Figure 7: A partial alignment created by SP71 when the longer sentences are presented before the shorter ones.

Given that the shorter sentences come before the longer ones, the program forms alignments like the one shown in Figure 8. From this it derives the patterns '$<$ 25 #34 f a s t $>$', '$<$ 24 #6 $<$ 1 $>$ $<$ 3 $>$ $>$' and '$<$ #35 $<$ 24 $>$ $<$ 25 $>$ $>$'. The second of these describe the structure of a short sentence and it contains the ID-symbol '24'. A copy of this symbol appears within the third pattern which means, in effect that the first pattern has been referenced from within the third pattern. As before, the program has recognised the intermediate level of structure and hierarchical relationship that is implicit in the original sentences.

```
0             w e           r u n   f a s t 0
              | |           | | |
1       < 1 #4 w e >        | | |           1
        | |       |         | | |
2       | |       | < 2 3 #5 r u n >        2
        | |       | |   |         |
3 < #6 < 1        > <   3         > >       3
```

Figure 8: A partial alignment created by SP71 when the shorter sentences are presented before the longer ones.

These intermediate results are reflected in the final grammars for all 12 sentences that are produced by the program. Very similar results are produced regardless of whether the shorter sentences are presented before the longer sentences or *vice versa*. Here, the results are described when the longer sentences are presented before the shorter ones. In this case, the second-best grammar produced by the program, after cleaning and tidying, is the grammar shown in Figure 9. This second-best grammar describes the structure of the longer sentences by means of the pattern '$<$ #1 $<$ 2 $>$ $<$ 4 $>$ $>$' that contains a reference, '$<$ 2 $>$' to the shorter sentence described by the pattern '$<$ 2 $<$ 1 $>$ $<$ 3 $>$ $>$'. The second-best grammar has captured the intermediate level of structure and the hierarchical relationship that is implicit in the original sentences. The best grammar is similar except

that it describes the structure of the shorter and longer sentences separately without any hierarchical reference from one to the other.

```
< 3 #3 w a l k >
< 1 #4 w e >
< 3 #2 r u n >
< 4 #7 f a s t >
< 1 #5 y o u >
< 5 #6 s l o w l y >
< 2 < 1 > < 3 > >
< #1 < 2 > < 4 > >
```

Figure 9: The second best grammar produced by SP71 when the longer sentences are presented before the shorter ones.

These results are in accordance with our expectations except that one might think that the best and second-best grammars are in the wrong order. It is not clear at this stage whether this ordering of the grammars is intrinsic to the data or whether it reflects a defect in the design or implementation of the program.

Alert readers will have spotted another defect in the grammar shown in Figure 9: the reference '$< 4 >$' at the end of the last pattern is appropriate for '< 4 #7 f a s t $>$' but not for '$<$ 5 #6 s l o w l y $>$'. This is clearly due to a deficiency in the program which should be remedied by further development.

8 Principles of Minimum Length Encoding

As successive New patterns are processed, SP71 records the values of G, E and $(G + E)$ for each grammar found. For the set of patterns shown in Figure 6, these values for the best grammar at each stage are plotted in Figure 10 together with the cumulative value of the sizes of the New patterns in unprocessed form (marked as O) and values for the overall compression achieved (calculated as $(G + E)/O$)).

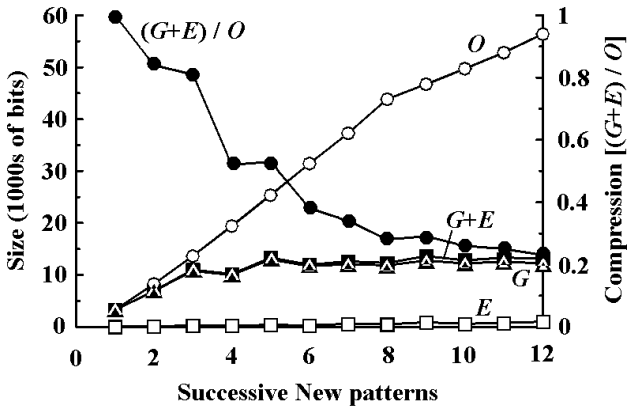

Figure 10: Plots showing changing values for G, E and $(G + E)$ for the best grammar found as successive New patterns are processed from the set shown in Figure 6. Other plots are described in the text.

It is clear from these graphs that, in accordance with principles of Minimum Length Encoding, the value of $(G + E)$ is quickly exceeded by the corresponding value of O and, accordingly, that values for compression fall steadily as learning proceeds.

9 Conclusion

The SP71 model has demonstrated significant success in abstracting intermediate levels of structure from appropriate input, as well as structures at the lowest and highest levels of abstraction. Further development and refinement is needed to overcome shortcomings in the model and the model must be tested on a wider range of inputs to establish its robustness and generality for the abstraction of plausible grammars from realistic data.

References

[Cover and Thomas, 1991] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. John Wiley, New York, 1991.

[Gold, 1967] M. Gold. Language identification in the limit. *Information and Control*, 10:447–474, 1967.

[Rissanen, 1978] J. Rissanen. Modelling by the shortest data description. *Automatica-J, IFAC*, 14:465–471, 1978.

[Solomonoff, 1964] R. J. Solomonoff. A formal theory of inductive inference. parts I and II. *Information and Control*, 7:1–22 and 224–254, 1964.

[Wallace and Boulton, 1968] C. S. Wallace and D. M. Boulton. An information measure for classification. *Computer Journal*, 11(2):185–195, 1968.

[Wolff, 2000] J. G. Wolff. Syntax, parsing and production of natural language in a framework of information compression by multiple alignment, unification and search. *Journal of Universal Computer Science*, 6(8):781–829, 2000. Copy: http://arxiv.org/abs/cs.AI/0307014.

[Wolff, 2002] J. G. Wolff. Unsupervised learning in a framework of information compression by multiple alignment, unification and search. Technical report, CognitionResearch.org.uk, 2002. Copy: http://arxiv.org/abs/cs.AI/0302015.

[Wolff, 2003a] J. G. Wolff. Information compression by multiple alignment, unification and search as a unifying principle in computing and cognition. *Artificial Intelligence Review*, 19(3):193–230, 2003. Copy: http://arxiv.org/abs/cs.AI/0307025.

[Wolff, 2003b] J. G. Wolff. Unsupervised grammar induction in a framework of information compression by multiple alignment, unification and search. In C. de la Higuera, P. Adriaans, M. van Zaanen, and J. Oncina, editors, *Proceedings of the Workshop and Tutorial on Learning Context-Free Grammars*, pages 113–124, 2003. This workshop was held in association with the 14th European Conference on Machine Learning and the 7th European Conference on Principles and Practice of Knowledge Discovery in Databases (ECML/PKDD 2003), September 2003, Cavtat-Dubrovnik, Croata. Copy: http://arxiv.org/abs/cs.AI/0311045.