Using Analogy

Ian. C. Stirk

Introduction

In a previous paper (Stirk 2001), I gave a rather abstract defence of the use of analogy in formal linguistics, showing that grammars based on analogy would put human languages into the class of mildly context sensitive languages, which seems correct. In what follows, I want to look at some of the practical possibilities of using analogy for the elucidation of how languages work.

Casting on

This is the term used for beginning a piece of knitting – forming the first loop to which all the others will be attached. I never mastered this with wool, but I hope to do a little better with starting off analogies and getting somewhere with them.

The simplest beginning I can think of for the English language is the two word sentence. Let us look at some :

marywalks
johnwalks
maryruns

Imagine an ideal language learner - an automaton rather than a human being - with a capacious memory. Any sentence containing new material is stored verbatim in the memory. We suppose that these are the first three sentences presented to the automaton, so that all three are entered into memory. Once they are in, though, the automaton can form a new sentence

johnruns

by using the analogy

marywalks : johnwalks :: maryruns : *johnruns*

Since the new sentence can be formed by analogy, there is no need to store it separately in memory. Of course English is being considered here, quite legitimately, as an orthographic language: phonetic symbols, or some other device, would have to be used to illustrate the automaton's acquisition of the spoken language. Word boundaries are not indicated, because they can be determined by the analogy process itself.

It is not very impressive so far that three sentences have to be committed to memory in order for just one more to be generated by analogy. But now suppose that we continue the process by presenting more sentences to the automaton after the three above. The results of the continuation can conveniently be expressed by means of a table :

	walks	runs	sings	dances
mary	1	3		
john	2		5	
sally		4		7
josephine			6	
gwilym				8
bert			9	
angharad		10		
jane	11			

Sentences 1, 2 and 3 are those already noted, while 5 to 11 are sentences containing new material, which need to be stored. The blank cells show sentences that can be formed by analogy from the 11, as the reader may readily check. They amount to 21 new sentences, building up the $8 \times 4 = 32$ cell matrix. Elementary mathematics shows that for an $m \times n$ matrix, $m + n - 1$ sentences need to be stored in memory, while the remainder can be generated analogically. The saving in memory storage thus grows rapidly as m and n increase. With 100 names and 100 verbs, the automaton would need to stock its memory with just 199 items to gain the ability to generate 9,801 other sentences.

Language pairs

For my next trick, I translate the 11 sentences of the matrix into Welsh :

1 maemaryyncerdded = marywalks
2 maejohnyncerdded = johnwalks
3 maemaryynrhedeg = maryruns
4 maesallyynrhedeg = sallyruns
5 maejohnyncanu = johnsings
6 maejosephineyncanu = josephinesings

7 maesallyyndawnsio = sallydances

8 maegwilymyndawnsio = gwilymdances

9 maebertyncanu = bertsings

10 maeangharadynrhedeg = angharadruns

11 maejaneyncerdded = janewalks

Clearly 21 new pairs can be formed analogically from these. There are two ways of doing this. In the first, the sentences in each language can be considered separately and then combined:

marywalks : johnwalks :: maryruns : *johnruns*

maemaryyncerdded : maejohnyncerdded :: maemaryynrhedeg : *maejohnynrhedeg*

In the second, the pairs are considered as one string :

maemaryyncerdded = marywalks : maejohnyncerdded = johnwalks ::
maemaryynrhedeg = maryruns : *maejohnynrhedeg = johnruns*

The purpose of the sentence pairs is that one can provide a semantics for the other. It is a liberal version of the method employed in the early days of Phrase Structure Grammar (see for instance Gazdar, Klein, Pullum and Sag, 1985, and the references therein) where each sentence is accompanied by its translation into the language of intensional logic. Translations into intensional logic could also be employed analogically, of course, but I think the flexibility of using another human language as the other member of the pair can be useful as well as interesting. In terms of the language learning automaton, we can imagine presenting it with one half of a pair, on which it will produce the other half: the equivalent in the other language.

To continue with Welsh-English pairs, suppose we add this one to our collection :

yrwyfiyngwybodfodjohnyncanu = iknowthatjohnsings

It contains new material, so it would have to occupy more space in the automaton's memory, but the results of analogising are quite impressive. The automaton could already produce 32 sentence pairs, and the addition of the new pair will double that number to 64, since "yrwyfiyngwybodfod" can replace the initial "mae" in any of the Welsh sentences, and "iknowthat" can prefix any of the English ones. The process can be imagined as adding a third dimension to the two dimensional grid of sentences that was used as an illustration above.

Adding the next pair

maeangharadynrhedegynaml = angharadoftenruns

would bring the total number of possible sentence pairs to 128, since any of the 64 previous ones could have "ynaml" and "often" added to them.

Adding some transitive sentences, such as

maejohnynadnabodmary = johnknowsmary
maeangharadyncarugwilym = angharadlovesgwilym
maejosephineyngweldbert = josephineseesbert
maejaneynclywedjohn = janehearsjohn
will add a large number of extra possible sentences to the total, which is not worth calculating precisely. In fact, it is probably clear that we have already passed the

threshold of an indefinite number of new sentences, the touchstone of any viable grammar since the earliest days of Chomsky. We can prefix "iknowthat" to any of the English sentences we already have, including those beginning with "iknowthat". In Welsh, the picture is slightly more complicated. We find the pair

iknowthatiknowthatjohnsings = yrwyfiyngwybodfymodiyngwybodfodjohnyncanu

showing that "thatiknow" translates as "fymodiyngwybod", involving a more substantial change than prefixing some items. But once this item is stored in memory, recursion can take place by analogy to give any number of Welsh sentences of increasing length.

Up to now we have been a bit cavalier about the nature of analogy, assuming it will always work to give us the results we already expected, as human language speakers. But it is all too easy for our human intuition to be used unconsciously, as the following attempted analogy may show:

maeangharadynrhedeg = angharadruns :
maeangharadynrhedegynaml = angharadoftenruns ::
maejohnyncanu = johnsings : maejohnyncanuynaml = johnoftensings

Are you convinced? The Welsh case is straightforward, as the adverb is just appended to the simple sentence, but in the English case the adverb is inserted between the subject and the verb, and this could cause difficulty to the unintuitive automaton. Why should "often" be inserted just where it is? Why not "jooftenhnsings", or any of the various other possibilities? This is a serious point, for if human intuition were allowed to slip into analogy making, we could hardly use the results to explain why human languages work as they do. The process of analogy making must be entirely

algorithmic.

A tentative analogy algorithm

Back to the simplest cases for a moment. An analogy like

marywalks : johnwalks :: maryruns : *johnruns*

just has to work, but how can we make it mechanical? Look at this way :

m	a	r	y	w	a	l	k	s
j	o	h	n	w	a	l	k	s
m	a	r	y	r	u	n	s	λ

j	o	h	n	r	u	n	s	λ

"λ" is used here to indicate a blank cell. Notice how the fourth row, the result of the analogy, can be formed column by column. The first three entries of the first column go "m-j-m" - a further "j" is needed to complete the symmetry. Similarly the second column, starting "a-o-a" needs another "o" for completion. In the fifth column, the pattern changes. The column starts "w-w-r", needing another "r" for symmetry. And so on. A "nothing" occurs in the last column, "s-s-λ-λ". The reader may check that similar symmetries will occur in other analogies, and be convinced by trying out a few that this is a regular pattern that has been uncovered here.

Let us try it out on a more complicated example. Arabic is complex morphologically, with its triliteral roots. For instance, we can compare the forms "katab — wrote" and "maktub — written", "hasab — calculated (past tense)".

Actually I have simplified these examples phonologically, but this makes no difference to the present consideration, which is that the three forms given so far make it appear that "mahsub – calculated (past participle)" should be generable analogically. And indeed the algorithm predicts so:

λ	λ	k	a	t	a	b
m	a	k	λ	t	u	b
λ	λ	h	a	s	a	b

which leads to a fourth row :

m	a	h	λ	s	u	b

The proposed algorithm seems to cope properly with that sort of complexity. Another challenge for it would be a change in order of items in a sentence. Let us consider an imaginary dialect of English in which all yes/no questions are formed by an inversion of subject and verb, so that the question corresponding to "johnwalks" would be "walksjohn". Putting those two sentences together with "johnprevaricates" should give "prevaricatesjohn" as a new analogically formed question form. The verbs are chosen here especially because of their difference in length. In fact the analogy algorithm copes in a satisfactory way:

j	o	h	n	w	a	l	k	s	λ	λ	λ	λ	λ	λ	λ	λ	λ	λ
λ	λ	λ	λ	w	a	l	k	s	λ	λ	λ	λ	λ	λ	λ	j	o	h	n
j	o	h	n	p	r	e	v	a	r	i	c	a	t	e	s	λ	λ	λ	λ

λ	λ	λ	λ	p	r	e	v	a	r	i	c	a	t	e	s	j	o	h	n

A further challenge cannot be met quite so easily. There are many instances of reduplication in natural languages, the simplest of which may be illustrated by the forms "È povero povero – he is very poor" and "È stanco stanco – he is very tired"

of Italian. Looking at these analogically would give us:

è	p	o	v	e	r	o	λ	λ	λ	λ	λ	λ
è	p	o	v	e	r	o	p	o	v	e	r	o
è	s	t	a	n	c	o	λ	λ	λ	λ	λ	λ

which would result in

è	s	t	a	n	c	o	p	o	v	e	r	o

Not quite right. There should be no difficulty in amending the algorithm to take care of this case, but I have not yet worked out a suitable notation. Perhaps numbering the repeated columns would do, as follows:

	1	2	3	4	5	6	1	2	3	4	5	6
è	p	o	v	e	r	o						
è	p	o	v	e	r	o						
è	s	t	a	n	c	o						

è	s	t	a	n	c	o	p	o	v	e	r	o

Anyway, now we can reexamine some of the casual analogies mentioned before. Sentences with "iknowthat" could expand as follows:

								j	o	h	n	s	i	n	g	s										
i	k	n	o	w	t	h	a	t	j	o	h	n	s	i	n	g	s									
								i	k	n	o	w	t	h	a	t	j	o	h	n	s	i	n	g	s

i	k	n	o	w	t	h	a	t	i	k	n	o	w	t	h	a	t	j	o	h	n	s	i	n	g	s

Here the λ's have been missed out for clarity. The Welsh forms also analogise correctly:

													y	r	w	y	f	i	y	n	g	w	y
y	r	w	y	f	i	y	n	g	w	y	b	o	d	f	y	m	o	d	i	y	n	g	w	y
													y	r	w	y	f	i	y	n	g	w	y

Only the first twenty five columns are illustrated here — readers can readily work out the remainder for themselves, as well as the concluding fourth line!

A potentially more serious problem was that of finding the proper place in "johnsings" to insert "often". The algorithm does not seem to help much, for, starting like this:

a	n	g	h	a	r	a	d	λ	λ	λ	λ	λ	r	u	n	s
a	n	g	h	a	r	a	d	o	f	t	e	n	r	u	n	s

we see that anything can be placed to the left and right of "often". Whatever it is, the analogy can still go through. There is some mystery here in the workings of the algorithm, which I have so far still not fathomed. Notice, though, that interestingly enough, the following analogy works fine:

a	n	g	h	a	r	a	d	r	u	n	s					
a	n	g	h	a	r	a	d	o	f	t	e	n	r	u	n	s
					j	o	h	n	r	u	n	s				

				j	o	h	n	o	f	t	e	n	r	u	n	s

The puzzling thing, of course, is that the analogy works unambiguously *precisely* because the letters of "runs" are *not* put into the same columns in the first and second rows. The meaning of this is still obscure to me. Anyway, having got as far as "johnoftenruns" we can proceed in a similar manner to "johnoftensings" :

					j	o	h	n	r	u	n	s	
j	o	h	n	o	f	t	e	n	r	u	n	s	
					j	o	h	n	s	i	n	g	s

j	o	h	n	o	f	t	e	n	s	i	n	g	s

Thus we can get from "angharadoftenruns" to "johnoftensings" unambiguously in two steps, via the intermediate form "johnoftenruns". Analogies in two steps also help with transitive sentences. As an example, let us derive "johnlovesjane" and its Welsh equivalent, "maejohnyncarujane" from "maeangharadyncarugwilym = angharadlovesgwilym", which is already in memory. Firstly we establish:

j	o	h	n					s	i	n	g	s						
a	n	g	h	a	r	a	d	s	i	n	g	s						
a	n	g	h	a	r	a	d	l	o	v	e	s	g	w	i	l	y	m

j	o	h	n					l	o	v	e	s	g	w	i	l	y	m

that is, "johnlovesgwilym". Next we go on to:

									j	a	n	e			d	a	n	c	e	s
									g	w	i	l	y	m	d	a	n	c	e	s
j	o	h	n	l	o	v	e	s	g	w	i	l	y	m						

j	o	h	n	l	o	v	e	s	j	a	n	e								

thus obtaining "johnlovesjane" via "johnlovesgwilym". The process in Welsh is similar:

m	a	e	j	o	h	n					y	n	c	a	n	u						
m	a	e	a	n	g	h	a	r	a	d	y	n	c	a	n	u						
m	a	e	a	n	g	h	a	r	a	d	y	n	c	a	r	u	g	w	i	l	y	m

m	a	e	j	o	h	n					y	n	c	a	r	u	g	w	i	l	y	m

and then :

m	a	e										j	a	n	e			y	n	d	a	w	n	s	i	o
m	a	e										g	w	i	l	y	m	y	n	d	a	w	n	s	i	o
m	a	e	j	o	h	n	y	n	c	a	r	u	g	w	i	l	y	m								

m	a	e	j	o	h	n	y	n	c	a	r	u	j	a	n	e										

When analogies go wrong

Here is a simple and short analogy, short enough for both languages to be analogised together :

j	a	n	e	w	a	l	k	s		=	j	a	n	e	p	i	e	d	i	r	a	s
j	a	n	e	w	a	l	k	e	d	=	j	a	n	e	p	i	e	d	i	r	i	s
j	a	n	e	s	w	i	m	s		=	j	a	n	e	n	a	g	h			a	s

Here the other language of the pair is Esperanto (without diacritics), chosen because of its regular morphology, as the result of the analogy is:

j	a	n	e	s	w	i	m	e	d	=	j	a	n	e	n	a	g	h			i	s

The Esperanto form of the past tense, "janenaghis", is quite correct, but the English

comes out as "janeswimed", rather than "janeswam". Nothing is wrong with the analogy process, so how do we handle irregular forms like this? In order to keep the apparatus of analogical grammar to a minimum, my suggestion here is to treat this as a case of new information : when the automaton comes up with an incorrect analogical form, the correct form is just added to memory. I hope that this may be enough to suppress the incorrect form, given a principle that in analogising, the automaton searches for the maximum amount of information in memory. Thus, asked to translate an Esperanto sentence containing the form "naghis", the automaton would find that in memory, with its counterpart "swam", and not bother to analogise from "nagh-" and "-is". That seems a reasonable expectation, but a little difficult to prove.

Prospects

Proofs, in fact, constitute the theoretical strand of future research into analogies. How do analogical grammars stand with respect to the Chomsky hierarchy, or, in other words, just how mild are the context sensitive grammars they determine? Another important theoretical point is to ensure that the method of dealing with irregularities, or failed analogies, will actually work.

It would probably be hard to disentangle that last point from empirical studies. It should be easy enough to implement the analogy algorithm in an actual computer program, and try out some actual language pairs just to see what happens. The results might throw up some new and interesting features.

The main point about the analogy method is its sheer simplicity. The hope is that with only the resources of a simple algorithm and a memory to contain any items that cannot be predicted by analogy, a human language can be encapsulated in a finite way. The example of "swam" shows that the details of a language can be captured,

and the broader picture can be made visible also. For an analogy may express the fact that, say, "jane" and "theladystandingoverthere" can be substitued one for the other, and thus belong to the same general class, although the class concept is not necessary for the algorithms to work.

The language pair concept is also rather flexible. A language of formal logic could be used for one member of the pair, or a semi-formal one might prove useful on occasion. Observe the following examples, for instance:

I know that John is reading the book

= I know that [John is reading the book]

I know that John is sitting in the chair

= I know that [John is sitting in the chair]

the chair John is sitting in is comfortable

= the chair is comfortable [John is sitting in the chair]

Here we have an adapted English as one language of the pair, using straight brackets to indicate subordinate clauses, and thus explicate something about relative clauses. Readers might like to discover for themselves what, if anything, may be analogised from those three sentence pairs!

The analogy concept also fits in well with at least one model of how language is handled in the human brain, that of Calvin and Bickerton (2000). They envisage a Darwinian process of sentence formation, in which previous memories of the usage of words play a role. The relation between this and analogies is obvious.

Conclusion

I hope I have been able to whet your appetite, as well as my own, for analogising. It has only been a matter of casting on so far, but I have great confidence in an interesting future for this strikingly simple method.

Bibliography

Calvin, W. H. and Bickerton, D. (2000) "Lingua ex Machina"
(MIT Press)

Gazdar, G., Klein, E., Pullum, G. and Sag, I. (1985)
"Generalized Phrase Structure Grammar"
(Basil Blackwell)

Stirk, Ian C. (2001) "Analogies and Universal Grammar"
(大阪外国語大学英米研究 Vol 25)

Ian C. Stirk, 2002